Une contribution à la résolution des processus décisionnels de Markov décentralisés avec contraintes temporelles

Aurélie Beynier

▶ **To cite this version:**

Aurélie Beynier. Une contribution à la résolution des processus décisionnels de Markov décentralisés avec contraintes temporelles. Informatique. Université de Caen, 2006. Français. NNT : . tel-00112014

HAL Id: tel-00112014
https://theses.hal.science/tel-00112014

Submitted on 7 Nov 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Département d'Informatique
UFR Sciences

École doctorale SIMEM

Une contribution à la résolution des Processus Décisionnels de Markov Décentralisés avec contraintes temporelles

THÈSE

présentée et soutenue publiquement le 13 novembre 2006

pour l'obtention du

Doctorat de l'Université de Caen Basse-Normandie

(spécialité Informatique)

par

Aurélie Beynier

Composition du jury

Directeur :	Mr. Abdel-Illah Mouaddib	Professeur	Université de Caen Basse-Normandie
Rapporteurs :	Mr. François Charpillet	Directeur de Recherche	LORIA
	Mr. Ranjit Nair	Senior Research Scientist	Honeywell Laboratories
	Mr. Shlomo Zilberstein	Professeur	University of Massachusetts Amherst
Examinateurs :	Mr. Rachid Alami	Directeur de Recherche	LAAS
	Mr. Olivier Sigaud	Professeur	Université Paris 6

Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen — CNRS UMR 6072

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier toutes les personnes qui m'ont permis de mener à bien cette thèse.

Mes remerciements s'adressent tout d'abord à Mr. Abdel-Illah Mouaddib, Professeur à l'Université de Caen, pour m'avoir accueillie dans son équipe et fait découvrir ce sujet de recherche passionnant. Je lui suis reconnaissante pour sa disponibilité et l'intérêt qu'il a porté à mon travail tout au long de ces trois années.

Je remercie également, pour leur investissement en tant que rapporteurs, Mr. François Charpillet, Directeur de Recherche à l'INRIA-Lorraine, Mr. Ranjit Nair, Chief Scientific Officer à GerminAIT Solution Pvt. Ltd., et Mr. Shlomo Zilberstein, Professeur à l'Université du Massachusetts. Un grand merci à Mr. Zilberstein pour l'accueil qui m'a été fait à Amherst et les échanges enrichissants que nous avons eus à propos de mon travail.

Je souhaite également remercier Mr. Rachid Alami, Directeur de Recherche au CNRS, et Mr. Olivier Sigaud, Professeur à l'Université Paris 6, pour avoir accepté de participer à mon jury de thèse et pour le temps qu'ils ont consacré à lire ce document.

Je tiens par ailleurs à saluer mes collègues au sein du GREYC et à remercier Laurent Jeanpierre pour le temps qu'il a accordé à mon travail et à la relecture de ce document. Nos longues discussions ont été très constructives.

Pour finir, mes pensées s'adressent à ma famille et mes amis. Je remercie Yann pour l'intérêt qu'il a toujours porté à mon travail, pour son soutien et sa patience. Enfin, je désire remercier tout particulièrement mes parents qui m'ont encouragée tout au long de mes études. Leur soutien et leur aide sont pour beaucoup dans l'accomplissement de cette thèse.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Résumé

Cette thèse porte sur la prise de décision distribuée dans des systèmes multi-agents agissant sous incertitude (les colonies de robots autonomes par exemple). Les processus décisionnels de Markov Décentralisés décrivent un formalisme mathématique permettant de modéliser et de résoudre de tels problèmes. Leur utilisation pour la planification des tâches dans des applications réelles pose toutefois quelques difficultés. Le modèle usuel des DEC-MDPs ne permet par exemple pas la prise en compte de contraintes sur l'exécution des tâches. De plus, la complexité de leur résolution est telle qu'il est difficile de déterminer une solution optimale excepté pour de petits problèmes.

Le travail que nous présentons dans cette thèse a pour premier objectif d'adapter le modèle des DEC-MDPs afin de proposer une modélisation adéquate du temps et des actions, et de permettre la représentation de problèmes réels. Nous décrivons ainsi une nouvelle classe de DEC-MDPs : les OC-DEC-MDPs (DEC-MDP avec Coût Occasionné). Dans un second temps, nous nous intéressons à leur résolution. Nous proposons différents algorithmes procédant à la planification des tâches de chaque agent en vue d'une prise de décision décentralisée et autonome, en accord avec les contraintes du problème. Afin de développer des algorithmes efficaces et de traiter des problèmes de taille importante, nous recherchons une approximation de la solution optimale. Nous procédons également à un découpage du problème initial en un ensemble de MDPs, et introduisons la notion de coût occasionné afin de tenir compte des interactions entre les agents et de calculer des politiques coopératives.

Introduction

L'idée de créer des êtres artificiels dotés d'intelligence est présente dans l'imaginaire des hommes depuis plusieurs milliers d'années. En effet, dès l'antiquité, différents mythes mettent en scène de telles entités déchargeant leur maître de tâches ingrates, fatigantes ou dangereuses comme le conte par exemple Homère dans l'Illiade à propos d'Héphaïstos[1]. Le mythe de Promethée et l'histoire du roi Pygmalion[2] reprennent eux aussi les thèmes de la créature artificielle toujours conçue par un dieu.

Jusqu'au XVI^{ème} siècle, cette idée du serviteur artificiel passe dans l'oubli. Elle réapparaît à Prague sous la forme du Golem, créature faite de boue, portée à la vie par une force occulte et chargée de réaliser toutes les tâches domestiques.

Au XIX^{ème} siècle, la révolution industrielle et les progrès scientifiques, en particulier l'invention de l'électricité[3], conduisent à penser qu'une telle créature peut naître de la science, sans l'intervention d'une force divine ou occulte. Ainsi, Mary Shelley [Shelley, 2005] écrit l'histoire d'un être créé de toutes pièces par le savant Dr Frankenstein. Le thème du scientifique découvrant le secret de fabrication d'un homme artificiel qui se révolte ensuite contre son concepteur, est également utilisé dans la pièce RUR (Rossum Universal Robots [Capek, 1997]) de Karel Capek où apparaît pour la première fois le terme de « robot ». Cette pièce innove en décrivant des êtres artificiels produits en masse. Elle donne également le ton aux récits de science fiction des années 1920 à 1930 où se multiplient les apparitions de robots belliqueux se retournant contre les hommes. En 1938, Isaac Asimov [Asimov, 2004], lassé de tels scénarios, écrit des histoires de robots réagissant à « *des règles logiques implantées dans leurs cerveaux* ». Il décrit alors des êtres artificiels au service de l'homme et dotés de qualités morales idéales.

[1] Héphaïstos aurait créé deux servantes en or chargées de l'aider à transporter les produits de sa forge jusqu'à l'Olympe.

[2] Pygmalion, roi de Chypre et sculpteur, réalisa une statue de son idéal féminin dont il tomba amoureux. En réponse à ses prières Aphrodite donna vie à la statue, la nommant Galatée.

[3] On pensait alors pouvoir redonner vie à des tissus morts grâce à l'électricité.

L'Intelligence Artificielle

L'apparition des premiers ordinateurs vers 1943 va permettre d'entrevoir la concrétisation de ce rêve. Les scientifiques vont alors chercher à concevoir des machines dotées d'intelligence conduisant ainsi à la naissance de l'intelligence artificielle (1956). Leur premier objectif est alors de développer des machines aussi intelligentes que l'homme. Après l'optimisme des premières années, les chercheurs vont prendre conscience de l'ampleur de la tâche et l'intelligence artificielle va être définie comme « l'élaboration d'entités intelligentes » [Russell et Norvig, 2003].

Toute l'ambiguïté de cette définition réside dans la caractérisation de la notion d'intelligence. Russell et Norvig [Russell et Norvig, 2003] proposent de classer les différentes définitions données à l'intelligence artificielle suivant qu'elles ont trait au processus de réflexion (raisonnement) mis en place ou au comportement affiché par le système développé. Parmi ces définitions, ils identifient alors les approches cherchant à développer des systèmes pensant ou agissant comme des humains. L'évaluation de l'intelligence de tels systèmes est bien entendu subjective et repose sur des vérifications empiriques. Un second point de vue consiste à définir un concept idéal d'intelligence : la rationalité. Russell et Norvig présentent alors l'intelligence artificielle comme l'élaboration d'entités agissant de manière rationnelle. Ils affirment d'ailleurs que le terme de « computational rationality » aurait été mieux adapté que celui d'intelligence artificielle. Ils argumentent que définir un système intelligent comme un système rationnel permet de s'abstraire des questions sur la nature de l'intelligence et sur les critères caractérisant l'intelligence d'une machine. Nous adopterons dans cette thèse un point de vue identique afin de définir l'intelligence d'un système. La rationalité étant un standard bien défini, la notion d'entité intelligente comme entité rationnelle sera ainsi clairement établie. Cette vision de l'intelligence artificielle présentera également l'avantage de n'imposer aucun mécanisme précis pour atteindre la rationalité. Nous reviendrons par la suite sur les critères de rationalité alors utilisés.

Notre domaine de recherche

Le domaine de l'intelligence artificielle peut être divisé suivant les problèmes abordés, ces derniers étant étroitement liés aux applications envisagées, ou bien suivant les techniques utilisées. Parmi les thèmes étudiés en intelligence artificielle nous pouvons citer : la démonstration automatique de théorèmes, les jeux, le traitement automatique du langage naturel (écrit ou oral, reconnaissance ou synthèse), la planification de tâches, la prise de décision, l'analyse d'images, la fouille de données, la robotique, ... Afin de développer des systèmes intelligents destinés à de telles applications, il est fait appel à une grande diversité de techniques : réseaux de neurones, approche heuristique, logique, modèles stochastiques, apprentissage, algorithme génétique ... Ces applications et techniques sont très nombreuses et sujettes à de nombreuses classifications dif-

férentes comme le suggère la grande diversité des catégorisations proposées par les conférences majeures portant sur l'intelligence artificielle.

Le travail que nous présentons dans cette thèse se situe à l'intersection de différents thèmes précédemment énumérés. Nous nous intéressons à la planification de tâches en vue de la prise de décision dans des colonies de robots autonomes, et plus généralement par des systèmes multi-agents.

Notre problématique

Comme nous venons de l'exposer, l'utilisation d'entités intelligentes en vue de la réalisation de tâches dangereuses, pénibles ou impossibles pour l'homme, n'est pas une idée nouvelle. Les récents progrès réalisés concernant le développement d'agents autonomes ont toutefois permis de concrétiser de tels projets auparavant cantonnés aux récits de science fiction.

Les recherches se sont tout d'abord orientées vers l'élaboration d'entités agissant seules (systèmes constitués d'un seul robot par exemple). Ces travaux ont démontré la faisabilité et les potentialités du développement de telles entités intelligentes. Plus récemment, les recherches en intelligence artificielle ont commencé à s'intéresser à la mise en interaction de telles entités et à la complexité des comportements qui en découle. C'est autour de cette idée que se sont développés les travaux concernant les systèmes multi-agents. Il est alors apparu que les décisions prises par une entité appartenant à un tel système sont complexifiées du fait des interactions. Il devient alors plus difficile, à partir des décisions individuelles, d'obtenir un comportement « collectivement intelligent ».

Ce problème va constituer le cœur de notre travail. Nous allons plus particulièrement chercher à élaborer un processus permettant à des entités de prendre, individuellement et de manière autonome, des décisions de façon à obtenir un comportement collectif rationnel. Les colonies de robots autonomes coopératifs constitueront l'application principale de notre travail. Nous développerons donc une approche permettant à chaque robot de prendre des décisions de manière autonome de façon à maximiser les performances de la colonie. Dans ce but, nous ferons appel aux Processus Décisionnels de Markov Décentralisés (DEC-MDPs), adaptation des Processus Décisionnels de Markov (MDPs) au cadre multi-agent.

Bien que les DEC-MDPs permettent la modélisation de problèmes décisionnels multi-agents, leur utilisation pour la résolution de problèmes concrets, par exemple pour la planification de tâches dans des colonies de robots autonomes, peut s'avérer difficile. Les DEC-MDPs ne permettent en effet pas la prise en compte de certaines données du problème comme les contraintes sur l'exécution des tâches. De plus, la complexité de leur résolution est telle qu'il est extrêmement difficile de calculer une solution optimale. Nous chercherons donc tout d'abord à adapter ce modèle afin de permettre la résolution de problèmes décisionnels multi-robots. Nous dévelop-

perons ensuite différents algorithmes permettant la planification des tâches de chaque entité en vue d'une prise de décision individuelle et autonome en accord avec les contraintes du problème. Nous chercherons alors à développer des algorithmes de résolution efficaces permettant la gestion de problèmes de taille importante tout en respectant les différentes contraintes sur l'exécution des tâches.

Plan synthétique de la thèse

Le présent document se décompose en quatre parties. La première partie réalise un état de l'art des différents outils utilisés par la suite dans nos travaux. La seconde partie introduit les problèmes auxquels nous nous intéressons et en propose une modélisation. La troisième partie porte sur la résolution des problèmes précédemment modélisés. Enfin, la quatrième et dernière partie présente les différentes expérimentations menées en vue de valider nos travaux.

Partie I : Décision sous incertitude dans les SMA

En nous appuyant sur une étude bibliographique, nous présenterons en première partie de ce document, les différents concepts et outils utilisés dans nos travaux. Un premier chapitre introduira les systèmes multi-agents. Nous présenterons tout d'abord le concept d'agent pour ensuite discuter de la mise en interaction de plusieurs de ces entités. Le second chapitre aura pour objet les Processus Décisionnels de Markov. Ces derniers proposent un formalisme mathématique particulièrement adapté à la résolution de problèmes décisionnels mono-agents sous incertitude. Ceci nous conduira à présenter, dans un troisième chapitre, une extension des MDPs au cadre multi-agent. Nous introduirons alors les DEC-MDPs et nous discuterons des difficultés liées à leur résolution.

Partie II : Un DEC-MDP pour la gestion de contraintes sous incertitude

La seconde partie de ce document s'articulera autour de deux chapitres. Le premier permettra, à partir du contexte applicatif de notre thèse, de définir clairement les problèmes auxquels nous nous intéressons. Nous identifierons alors les types de contraintes portant sur l'exécution des tâches dont il sera nécessaire de tenir compte par la suite. Le second chapitre portera sur la modélisation de ces problèmes à l'aide de DEC-MDPs. Après avoir présenté les difficultés liées à l'utilisation d'un tel formalisme, nous introduirons un nouveau modèle adapté à la représentation de contraintes : l'OC-DEC-MDP. Nous disposerons alors d'une modélisation adaptée aux problèmes qui nous sont posés.

Partie III : Résolution d'un OC-DEC-MDP

La troisième partie de ce document portera sur la résolution des problèmes précédemment modélisés. Nous développerons un algorithme efficace de planification des tâches permettant de tenir compte des différents types de contraintes émanant du problème. Le premier chapitre de cette partie permettra de finaliser la modélisation du problème sous forme d'OC-DEC-MDP. Nous présenterons ensuite, dans un second chapitre, différentes mesures permettant à un agent d'évaluer individuellement ses décisions en tenant compte de leur impact sur les autres agents. Nous introduirons alors les notions d'utilité espérée et de coût occasionné. Ces mesures nous permettront d'aboutir à un comportement collectivement rationnel. Elles seront ainsi utilisées dans les chapitres suivants décrivant chacun un algorithme de résolution des OC-DEC-MDPs. Le troisième chapitre de cette partie présentera un algorithme permettant de déterminer une approximation de la politique optimale par amélioration d'un ensemble de politiques initiales. Enfin, le quatrième chapitre décrira une version itérative du précèdent algorithme développée en vue d'améliorer la qualité des solutions retournées par notre approche.

Partie IV : Validation de l'approche

La quatrième partie de cette thèse s'attachera à démontrer la validité de nos travaux. Un premier chapitre s'intéressera à l'efficacité des algorithmes développés ainsi qu'à la qualité des solutions obtenues. Un second chapitre traitera plus particulièrement de l'applicabilité de nos travaux à des problèmes réels liés au développement de colonies de robots autonomes.

Nous conclurons enfin en tirant le bilan des travaux exposés dans cette thèse et présenterons différentes perspectives ouvertes par ce travail.

Les preuves des différents théorèmes et corollaires énoncés dans ce document sont détaillées en annexe. Une présentation succincte des robots Koala, utilisés lors de la validation de notre approche, est également jointe.

Première partie

Décision sous incertitude dans les SMA

Introduction de la partie I

Dans les pages qui précèdent, nous avons introduit l'Intelligence Artificielle comme ayant trait à la conception d'entités intelligentes. Nous avons choisi la rationalité comme concept pour définir l'intelligence d'un système. Nous pouvons ainsi envisager l'Intelligence Artificielle comme la conception d'agents rationnels. Avant d'aller plus loin dans la présentation de notre travail, il nous semble nécessaire de définir plus précisément un certain nombre de notions telles que la notion d'agent et de rationalité.

L'objet de cette première partie est donc d'introduire les concepts qui seront à la base du travail que nous présenterons par la suite. Nous commencerons, dans un premier chapitre, par définir la notion d'**agent** puis, nous aborderons la dimension sociale de ces entités et nous discuterons de leur mise en interaction. Les **systèmes multi-agents** seront alors présentés. Nous soulignerons l'importance du processus de décision dans la mise en place d'entités intelligentes et nous décrirons les problèmes liés à la décision dans les systèmes multi-agents.

Les deux chapitres suivants porteront sur la formalisation des problèmes décisionnels dans des environnements stochastiques. Le chapitre 2 introduira les **processus décisionnels de Markov** pour la résolution des problèmes de décision mono-agent. Le troisième et dernier chapitre de cette partie établira le lien entre les deux chapitres qui auront précédé. Nous nous intéresserons à l'adaptation des processus décisionnels de Markov dans un contexte multi-agent. Nous décrirons alors plus particulièrement les **processus décisionnels de Markov décentralisés** et nous aborderons les difficultés posées par leur résolution.

Chapitre 1

Les systèmes multi-agents

Bien que l'Intelligence Artificielle ait fait l'objet de nombreuses définitions et qu'elle puisse être abordée selon différents points de vue, il est possible de s'accorder sur le fait qu'elle consiste en la mise en place de systèmes intelligents également appelés agents.

La notion d'agent occupe donc une place prépondérante en Intelligence Artificielle et va constituer la base de ce premier chapitre. Nous commencerons par présenter la diversité des définitions du terme d'agent et nous développerons les concepts d'**environnement** et d'**autonomie** qui y sont étroitement liés. Nous décrirons ensuite les différentes caractéristiques d'un agent, en nous attardant plus particulièrement sur les agents rationnels. Comme nous l'avons souligné précédemment, nous adopterons alors le point de vue de Russell et Norvig.

Dans un second temps, nous nous intéresserons à la dimension sociale des agents qui constitue le propos de l'Intelligence Artificielle Distribuée. Nous introduirons les systèmes multi-agents et discuterons de la mise en interaction de plusieurs agents et de leur organisation. Nous reviendrons enfin sur certaines propriétés des systèmes multi-agents qui occuperont une place essentielle dans la suite de notre travail.

1.1 Présentation du concept d'agent

Dans cette première section, nous commencerons par présenter différentes définitions et caractéristiques des agents, pour ensuite préciser ce que nous désignerons par le mot agent dans le reste de notre travail.

1.1.1 La multiplicité des définitions

Intéressons nous tout d'abord à une définition très générale du mot « agent » donnée dans le dictionnaire *Le petit Robert* de la langue française.

Définition 1 ***Définition du mot agent selon le dictionnaire Le petit Robert***

- *L'être qui agit.*
- *Ce qui agit, opère ; force, corps, substance intervenant dans la production de certains phénomènes.*

Comme l'évoque cette définition, l'action est au cœur du concept d'agent. Le mot agent provient d'ailleurs du latin *agere* qui signifie « faire, mettre en mouvement ». Cette dimension de l'action est également à la base de la définition de Russell et Norvig [Russell et Norvig, 2003] présentée dans leur ouvrage introduisant les principes fondamentaux de l'intelligence artificielle.

Définition 2 ***Agent selon Russell et Norvig [Russell et Norvig, 2003]***

« An agent is just something that acts. » [4]

Bien que le mot agent soit souvent associé à un être humain (agent de change, agent de police, ...), il peut également désigner des notions bien plus abstraites comme nous pouvons le voir dans des expressions telles que « agent chimique, physique », « agent de contagion », « agent mutagène », ...

En ce qui nous concerne, nous nous intéresserons plus particulièrement aux agents informatiques[5] qui font l'objet de nombreuses études en intelligence artificielle. La notion d'agent en informatique a donné lieu à de nombreuses définitions dont Franklin et Graesser donnent un bref aperçu dans [Franklin et Graesser, 1996]. A l'heure actuelle, aucune définition n'a été universellement acceptée bien que certaines rassemblent la majorité de l'opinion. L'étude de ces définitions permet toutefois d'en identifier des traits communs.

La première question qui se pose lorsque nous parlons d'agents en informatique consiste à établir une différenciation entre un agent et un programme informatique. En effet, selon la définition précédente de Russell et Norvig, un agent est quelque chose qui agit. Dans ce cas, nous pouvons supposer qu'un programme informatique est un agent : un programme reçoit des informations en entrée et produit un résultat qui est retourné à la fin de son exécution. Un programme agit et pourrait donc être assimilé à un agent. Il existe cependant plusieurs notions fondamentales qui distinguent un simple programme d'un agent. En effet, Russell et Norvig ajoutent à leur définition : « *But computer agents are expected to have other attributes that distinguish them from mere programs, such as operating under autonomous control, perceiving environment, persisting over a prolonged period, adapting to change, and being capable of taking on another's goals* » [6].

[4] Un agent est simplement quelque chose qui agit.

[5] Russell et Norvig les désignent par le terme « computer agent »

[6] Cependant, les agents informatiques sont supposés posséder d'autres caractéristiques qui les distinguent de simples programmes, comme l'autonomie, la perception de l'environnement, la persistance sur une période prolongée, l'adaptation au changement, la faculté de modifier ses buts.

Ces notions d'autonomie, d'environnement et d'objectif, sont également présentes dans la définition de Wooldridge et Jennings [Wooldridge et Jennings, 1995] :

Définition 3 ***Agent selon Wooldridge et Jennings [Wooldridge et Jennings, 1995]***

« An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives. » [7]

Par perception de son environnement, un agent peut donc « récolter » des informations qu'il utilisera ensuite afin de décider de façon autonome comment agir. L'exécution d'une action ainsi que la dynamicité de l'environnement modifient les perceptions de l'agent qui décide alors d'une nouvelle action, et ainsi de suite. Cette boucle perception-action, représentée sur la figure 1.1, met en évidence les interactions entre l'agent et son environnement.

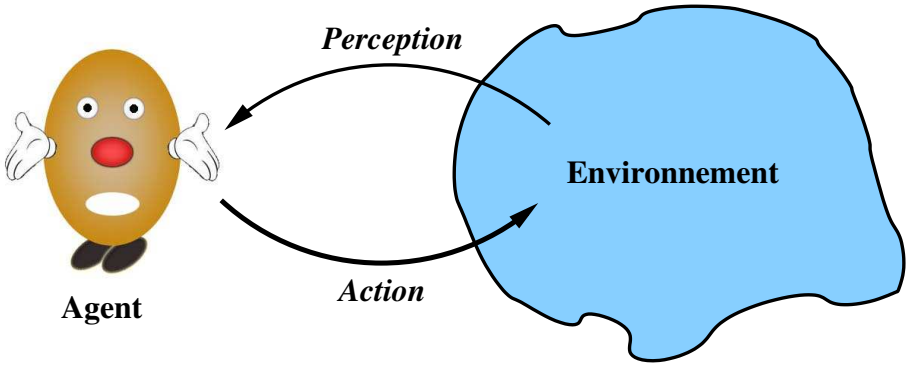


FIG. 1.1 – Agent et environnement

Ferber [Ferber, 1995] propose également une définition de la notion d'agent dans laquelle nous retrouvons ces concepts clefs. Sa définition est cependant plus orientée vers les systèmes multi-agents qui constituent la base de son travail.

Définition 4 ***Agent selon Ferber [Ferber, 1995]***

« Un agent est une entité physique ou virtuelle

- *qui est capable d'agir dans un environnement,*
- *qui peut communiquer directement avec d'autres agents,*
- *qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),*
- *qui possède des ressources propres,*
- *qui est capable de percevoir (mais de manière limitée) son environnement,*

[7] Un agent est un système informatique situé dans un environnement et capable d'agir de manière autonome dans cet environnement afin d'atteindre les objectifs pour lesquels il a été conçu.

- *qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),*
- *qui possède des compétences et offre des services,*
- *qui peut éventuellement se reproduire,*
- *dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit. »*

Cette définition évoque les concepts de compétence, objectifs, perception limitée et communication que nous décrirons par la suite. Ferber définit l'agent comme une entité physique ou virtuelle ce qui permet de mettre en évidence le fait qu'un agent n'est pas seulement restreint à un composant logiciel ou à un module informatique (virtuel). Il peut également s'agir d'un robot, d'un satellite, d'un avion, d'un aspirateur, d'un char d'assaut, etc.

L'agent informatique le plus basique qui soit décrit dans la littérature, consiste en un thermostat qui régule par lui-même la température d'une pièce [Wooldridge, 2002]. Le thermostat peut, à partir de ses capteurs, percevoir la température de la pièce. En fonction de ses perceptions, il décide de manière autonome comment agir sur le chauffage. A l'opposé, des agents d'une complexité beaucoup plus importante peuvent être envisagés, tels que les voitures autonomes pouvant s'insérer et évoluer dans une circulation dense.

1.1.2 Les concepts communs

A partir des définitions que nous avons présentées, nous pouvons mettre en évidence le fait que la définition d'un agent s'articule autour de plusieurs concepts communs en particulier, l'**agent** lui même, l'**environnement** et l'**autonomie**. Nous allons à présent revenir plus en détail sur chacun de ces trois concepts.

Agent

Toutes les définitions du concept d'agent informatique, s'accordent sur le fait qu'un agent est équipé de **capteurs** lui permettant de percevoir son environnement et d'**effecteurs** lui permettant d'agir dans cet environnement. A partir de ces capteurs, l'agent acquiert des connaissances sur son environnement qui peuvent enrichir des connaissances fournies *a priori* par le concepteur. Ces informations lui permettent alors de décider comment agir. Cette décision résulte d'un processus délibératif plus au moins complexe. Un agent peut, par exemple, baser ses décisions sur ses perceptions courantes, ou bien sur l'historique de ses perceptions. Après délibération, l'agent exécutera, grâce à ses effecteurs, l'action qu'il a décidé de réaliser.

Les types de capteurs et d'effecteurs que possède un agent sont étroitement liés à l'application pour laquelle il sera utilisé. Ainsi, un robot explorateur sur Mars sera équipé d'une caméra, d'un

thermomètre, d'un sonar, d'une boussole, etc. Il aura pour effecteurs des roues, un bras articulé, des pinces, une foreuse, ... A l'opposé, un agent professeur d'anglais aura pour capteurs un clavier et une souris, et pour effecteurs un écran et des hauts parleurs lui permettant de poser des questions et de donner des réponses.

L'ensemble des actions exécutables par un agent dépend des effecteurs à sa disposition. Certaines actions ne peuvent être exécutées que dans des situations précises, elles requièrent alors qu'un ensemble de pré-conditions soit satisfait.

Les capteurs et les effecteurs d'un agent peuvent être imprécis. De ce fait, l'information perçue par les capteurs peut s'avérer inexacte ou incomplète. Prenons l'exemple des capteurs infrarouges équipant les robots Koala que nous avons utilisés au cours de notre travail (cf. Annexe B). Ces capteurs permettent de détecter des obstacles par mesure de la lumière ambiante et de la lumière réfléchie. Cependant, la distance de détection varie de façon importante suivant les conditions d'éclairage et le pouvoir réfléchissant de l'obstacle. Ainsi, un robot détectera une personne habillée en noir à moins de 10 cm, un mur blanc moyennement éclairé est détecté à environ 50 cm et une bouche d'aération en aluminium brossé à plus d'un mètre ! De plus, les mesures réalisées dans un environnement éclairé par des néons seront fortement dépendantes de la fréquences de ces derniers[8]. Ces capteurs infrarouges ne permettent donc pas d'avoir une représentation complète et exacte de l'environnement.

En diversifiant et en multipliant les capteurs d'un agent, il est possible, dans certains cas, d'augmenter l'information perçue et son exactitude. Nous pouvons par exemple ajouter aux robots Koala un sonar afin d'améliorer ses perceptions. Nous nous trouvons cependant confrontés à d'autres problèmes de perception tels que les échos multiples et la détection de surfaces poreuses. Par ailleurs, la multiplication des capteurs nécessite de fusionner des informations en provenance de différentes sources, ce qui n'est pas sans poser de problèmes, les informations pouvant s'avérer contradictoires ou d'échelles différentes.

En ce qui concerne les effecteurs, des problèmes d'imprécision peuvent également se poser. En effet, le résultat de l'exécution d'une action n'est pas totalement contrôlé par l'agent, il dépend également des conditions dans lesquelles l'action est réalisée. Wooldridge parle de « contrôle partiel » [Wooldridge, 2002] puisque l'agent n'est pas totalement maître du résultat de ses actions. Reprenons l'exemple des robots Koala, et supposons que nous souhaitions que le robot décrive un carré. Suivant l'inclinaison du sol, l'alignement des roues, la texture du sol, etc., le robot va plus ou moins dévier. De ce fait, après avoir décrit le carré, la probabilité que le robot revienne exactement à sa position initiale est faible. Bien que certaines déviations puissent se trouver compensées par d'autres, de manière générale, le robot s'arrêtera dans un certain périmètre autour de son point de départ. La figure 1.2 décrit le déplacement théorique du robot et donne l'exemple d'un déplacement réel. Nous constatons que le robot ne revient pas exactement à

[8] Au grand dam des concepteurs de programmes !

son point de départ. L'amplitude de la déviation est très dépendante de l'environnement, des mouvements et du calibrage des robots, il est donc difficile de l'estimer *a priori*. A titre indicatif, sur un sol plastique lisse, nous avons pu constater des déviations de l'ordre de dix degrés sur des déplacements en ligne droite.

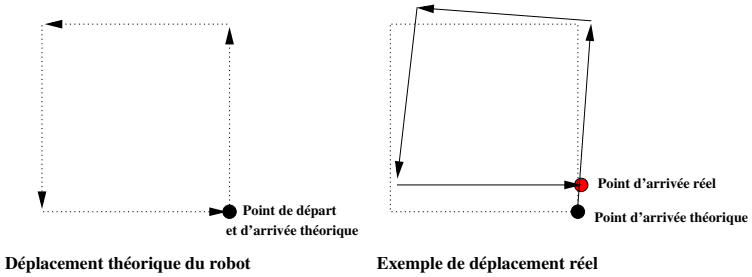


FIG. 1.2 – Déviation lors du déplacement d'un robot

Des problèmes d'incertitude sur l'exécution peuvent également être envisagés dans l'exemple de l'agent « thermostat ». Lorsque la température est trop basse et que l'agent augmente le chauffage, il n'est pas certain que la température de la pièce augmente. En effet, l'action peut s'avérer inefficace si la porte ou la fenêtre sont ouvertes.

L'imprécision des capteurs et des effecteurs permet de mettre en évidence deux types d'incertitude auxquels un agent doit faire face : le premier concerne l'observabilité de l'environnement et le second se rapporte au résultat des actions. Ces incertitudes sont à mettre en relation avec certaines caractéristiques de l'environnement, en particulier l'observabilité et l'aspect déterministe.

Environnement

Russell et Norvig [Russell et Norvig, 2003] ont identifié un certain nombre de propriétés permettant de caractériser l'environnement d'un agent. Un environnement peut être :

- **totalement observable ou partiellement observable**[9] : Un environnement est dit totalement observable si l'agent perçoit toutes les informations nécessaires à la décision. Une telle observabilité est souvent impossible en raison de l'imprécision ou des limitations des capteurs. L'agent ne perçoit alors qu'une partie de l'environnement qui est dit partiellement observable. Dans les cas extrêmes, l'agent peut ne rien percevoir.
- **déterministe ou stochastique** : Lorsque le prochain état de l'environnement est totalement déterminé par son état courant et par l'action exécutée, l'environnement est dit déterministe. Nous avons alors $P(s, a, s') \in \{0, 1\}$ où $P(s, a, s')$ est la probabilité de passer d'un état s à un état s' quand l'agent exécute l'action a. Si à partir d'un même état et

[9] On parle également d'environnement accessible ou inaccessible.

d'une même action, il est possible d'arriver dans différents états alors, l'environnement est stochastique. D'où $P(s, a, s') \in [0, 1[$. Un environnement déterministe peut paraître stochastique du point de vue de l'agent si celui-ci n'en a qu'une connaissance partielle. Etant donné qu'il est difficile de modéliser parfaitement un environnement, celui-ci est généralement supposé comme étant stochastique. Ceci permet de représenter, par une distribution de probabilités, l'incertitude sur les aspects inconnus on non modélisés de l'environnement.

- **épisodique ou séquentiel** : Dans un environnement épisodique, l'expérience de l'agent peut être divisée en « épisodes ». Chaque épisode consiste en une phase de perception, suivie de l'exécution d'une seule action. Chaque épisode est indépendant des autres, le résultat de l'exécution d'une action ne dépend que de l'épisode courant. Dans un environnement séquentiel, l'exécution d'une action peut influencer toutes les actions à venir.
- **statique ou dynamique** : Si l'environnement est susceptible de changer pendant la phase de délibération de l'agent, alors nous avons à faire à un environnement dynamique. Sinon, il est statique. Dans un environnement statique, l'agent n'a pas à se soucier du fait que l'environnement peut avoir changé depuis qu'il a commencé à délibérer et que, par conséquent, sa décision soit devenue inappropriée.
- **discret ou continu** : Un environnement est discret s'il existe un nombre fini de perceptions et d'actions possibles. Sinon il est continu.

Les environnements les plus complexes sont les environnements partiellement observables, stochastiques, séquentiels, dynamiques et continus. De tels environnements sont dits ouverts [Hewitt, 1986]. Les caractéristiques d'un environnement influencent la complexité de la conception de l'agent. Moins un agent dispose d'information, plus il lui est difficile de prendre une bonne décision. Wooldridge [Wooldridge, 2002] argumente que la qualité de la décision dépend de la qualité de l'information disponible. Plus l'agent aura d'informations fiables pour prendre une décision, plus celle-ci s'avérera adaptée à la situation. Ainsi un agent devant agir dans un environnement partiellement observable devra faire face à un manque d'information « fiable » qui complexifiera la prise de décision, par rapport à un agent ayant une connaissance complète de son environnement.

Autonomie

Considérons un agent placé dans un environnement et devant décider, à partir de ses perceptions, quelle action exécuter. Si cette décision est prise sans l'intervention d'un tiers, l'agent est dit **autonome**. Cette définition de l'autonomie, donnée par Wooldridge et Jennings [Wooldridge et Jennings, 1995], diffère quelque peu de la définition de Russell et Norvig [Russell et Norvig, 2003].

Selon Russell et Norvig, l'autonomie caractérise un agent capable de s'affranchir des connaissances de son concepteur. Ainsi, un agent autonome base ses décisions uniquement sur ses connaissances acquises, elles ne requièrent pas un savoir fourni *a priori* par le concepteur. Un tel agent doit donc être en mesure d'acquérir des connaissances afin de pallier son manque de savoir initial. Son comportement s'améliorera au cours de l'expérience qui lui permettra d'engranger des connaissances. Des techniques d'apprentissage sont alors mises en œuvre afin d'améliorer le comportement de l'agent.

Tout comme les propriétés de l'environnement, le degré d'autonomie influence la complexité de la conception d'un agent. En effet, si nous souhaitons développer un agent qui puisse s'affranchir de l'intervention d'un tiers (que ce soit à la conception ou à l'exécution), il est nécessaire de mettre en place des mécanismes décisionnels flexibles qui lui permettent de faire face tout seul à un grand nombre de situations.

Quelle que soit la définition du mot « agent » considérée, nous pouvons remarquer que les concepts d'agent et d'environnement sont étroitement liés : tout agent est situé dans un environnement qu'il perçoit et dans lequel il agit. L'autonomie quant à elle est une caractéristique majeure des agents qui permet de les différencier des simples programmes informatiques.

Nous allons à présent passer en revue un certain nombre de propriétés des agents qui permettront de définir plus précisément le type d'entités auxquelles nous nous intéresserons par la suite.

1.1.3 Les propriétés d'un agent

Les propriétés qui vont suivre influencent essentiellement le comportement de l'agent, elles conditionnent son « degré d'intelligence ». Elles auront donc, tout comme l'autonomie ou les caractéristiques de l'environnement, un impact sur la manière de concevoir l'agent.

Rationalité

La **rationalité** caractérise ce qui est raisonnable, fait avec bon sens. Ce concept, à la base des théories économiques, est également largement étudié dans des domaines tels que la sociologie, les sciences de la gestion, les sciences politiques, la psychologie cognitive, etc.

Un **agent rationnel** est un agent qui agit « au mieux », il prend à tout moment la meilleure décision. Nous pouvons alors nous interroger sur ce qu'on entend par « agir au mieux » ou « la meilleure décision ». Mathématiquement, la recherche de ce qui est « meilleur » ou « optimal », se traduit par l'optimisation d'une fonction caractérisant les objectifs de l'agent. Ceci nous amène à introduire la notion de **mesure de performance**. Cette dernière permet d'évaluer une décision en terme de succès. La décision maximisant la mesure de performance est celle qui va garantir

le plus de succès à l'agent, elle est alors appelée décision rationnelle.

La mesure de performance prend en compte les connaissances de l'agent, ses perceptions et les actions exécutées. Russell et Norvig définissent alors un agent rationnel de la façon suivante :

Définition 5 ***Agent rationnel selon Russell et Norvig [Russell et Norvig, 2003]*** *« For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has. »* [10]

La mise en place d'une mesure de performance adaptée est un problème difficile : elle doit tenir compte des actions immédiates mais aussi de leurs effets à long terme. Par ailleurs, la rationalité d'un agent s'avère être un critère subjectif dépendant du concepteur. Suivant le comportement qu'il souhaite que son agent adopte, le concepteur définira une mesure de performance plutôt qu'une autre. Un comportement irrationnel d'un certain point de vue peut paraître tout à fait rationnel d'un autre.

Herbert Simon, prix Nobel d'économie en 1987, s'est intéressé à la rationalité dans de nombreux domaines, parmi lesquels l'intelligence artificielle. Il a mis en évidence l'importance majeure des perceptions et des connaissances dans la prise de décision. Il a également étudié les limites de la rationalité des décideurs et a introduit la distinction entre rationalité limitée et rationalité optimisante [Simon, 1947]. La **rationalité optimisante** (appelée également objective ou parfaite) étudiée jusqu'alors, supposait que le décideur disposait d'une connaissance parfaite de toutes ses possibilités et des conséquences de ses décisions, son choix ne pouvait alors qu'être optimal. Selon Simon « *il est impossible que le comportement d'un seul individu isolé atteigne un degré de rationalité élevé. Le nombre d'alternatives à explorer est si grand, l'information nécessaire à leur évaluation si vaste que même une approximation de la rationalité objective est difficile à concevoir* ». Simon objecte donc que les capacités des décideurs sont restreintes, c'est pourquoi tous les éléments pertinents au regard de la décision ne peuvent être envisagés. Le choix optimal ne peut alors être déterminé et l'agent doit se contenter de chercher une solution satisfaisante, on parle de **rationalité limitée**.

De nombreux travaux font l'hypothèse d'une rationalité optimisante désignée par Russell et Norvig par le terme de « rationalité parfaite ». Il est alors supposé qu'à chaque instant l'agent sera en mesure de maximiser sa mesure de performance étant données les informations qu'il

[10] Pour chaque séquence possible de perceptions, un agent rationnel devrait sélectionner l'action pour laquelle il s'attend à maximiser sa mesure de performance, étant données les preuves fournies par sa séquence de perceptions et quelles que soient ses connaissances *a priori*.

aura obtenues de son environnement (l'agent peut toutefois avoir une observabilité partielle). Cependant, dans des environnements complexes de tels calculs peuvent rarement être envisagés. Il est alors nécessaire de s'orienter vers la recherche d'un certain degré de satisfaction menant à une rationalité limitée.

Réactivité

La **réactivité** est la capacité d'un agent à percevoir son environnement et à décider comment agir en respectant le temps qui lui est imparti. Un tel agent est en mesure de prendre ses décisions en temps réel.

Les agents dits « réactifs » basent leur comportement sur un ensemble de stimuli-réponses. Chaque stimulus (ou situation) fait correspondre une réponse (ou action). Ce processus de délibération étant simple à appliquer, ces agents font preuve d'une grande réactivité, d'où leur nom d'agents réactifs. Cependant, des agents ayant un processus de perception et de délibération plus complexe peuvent également montrer une forte réactivité. Nous préférerons donc utiliser l'appellation d'agents réflexes[11] de Russell et Norvig afin de désigner les agents dont le processus de délibération est basé sur des règles stimulus → réponse.

Le temps de réponse octroyé à un agent pour percevoir son environnement et décider quelle action réaliser est dépendant du système. Les exigences concernant ce délai peuvent donc différer d'une application à une autre. Si nous souhaitons qu'un agent réalise un maximum de tâches en un minimum de temps, nous chercherons à minimiser le temps de réaction des agents et ainsi obtenir une forte réactivité. De manière générale, plus un mécanisme délibératif sera simple, plus l'agent fera preuve d'une grande réactivité.

Degré de délibération

Tout agent reçoit un ensemble de perceptions qui le conduit à exécuter des actions. Le passage des perceptions aux actions est réalisé par un **processus délibératif** qui permet à l'agent de décider quelle action exécuter. Suivant les spécificités des applications, ce processus délibératif peut s'avérer plus ou moins complexe. En s'appuyant sur les propos de Russell et Norvig [Russell et Norvig, 2003], nous allons présenter différents processus délibératifs possibles. Nous commencerons par le plus simple pour ensuite le complexifier au fur et à mesure de la mise en évidence des lacunes des modèles proposés.

Nous avons déjà décrit les agents réflexes dont le comportement est basé sur des règles stimulus → réponse. Ces agents possèdent un mode de délibération des plus simples mais leur intelligence est « limitée ». En effet, la décision d'un tel agent est uniquement basée sur ses perceptions courantes, ce qui peut conduire à des comportements totalement inappropriés en cas de

[11] Simple reflex agents

manque d'observation : l'agent ne perçoit pas une information cruciale qui le conduit à prendre une mauvaise décision.

Il est possible de rectifier ce problème en permettant aux agents de construire une représentation interne de leur état résumant l'ensemble de leurs perceptions passées et de leurs connaissances *a priori*. Cet état constitue une représentation du monde beaucoup plus précise que les seules perceptions courantes de l'agent. L'agent est alors en mesure de pallier un manque d'observation immédiat, ce qui le conduit à un comportement plus en adéquation avec la situation.

Bien que cette représentation interne améliore les performances des agents, elle n'est pas suffisante dans bien des cas. En effet, elle limite l'agent à une vision à court terme de ses actions alors qu'il est généralement nécessaire que l'agent puisse en envisager les conséquences à long terme. La **pro-activité** caractérise la capacité d'un agent de se fixer des buts afin d'atteindre des objectifs. Un agent pro-actif peut ainsi planifier ses actions afin de satisfaire ses buts. Il n'envisage plus seulement les effets immédiats de ses actions. Notons que les buts d'un agent peuvent changer au fil de l'exécution des actions. Si les modifications de l'environnement rendent ses buts inatteignables, l'agent doit être en mesure de définir de nouveaux buts et de replanifier ses actions.

Comme l'affirment Russell et Norvig, l'orientation du comportement d'un agent par des buts n'est pas suffisant pour que celui-ci ait un comportement de haute qualité. Elle permet de définir si un comportement est souhaité ou non, c'est-à-dire s'il permet d'atteindre les buts fixés ou non. Cependant, il n'est pas possible d'établir un ordre de préférence entre les comportements souhaités. Supposons que nous voulions nous rendre de notre bureau à la gare. Il existe différents trajets permettant d'atteindre notre but qui est la gare. Néanmoins, nous pouvons préférer le plus court chemin ou bien le plus rapide, et ainsi établir un classement des trajets possibles. Un tel ordre de préférence peut être exprimé par la notion d'utilité.

Une **fonction d'utilité** définit des préférences sur les états. Elle permet donc d'associer à chaque état s un nombre réel décrivant le degré de satisfaction de l'agent lorsqu'il est dans s.

La théorie de l'utilité espérée trouve ses racines en économie et repose sur le théorème de Von Neumann et Morgenstern présenté en 1944 dans un ouvrage intitulé *Theory of Games and Economic Behavior* [Von Neumann et Morgenstern, 1944]. Ce théorème considère un ensemble

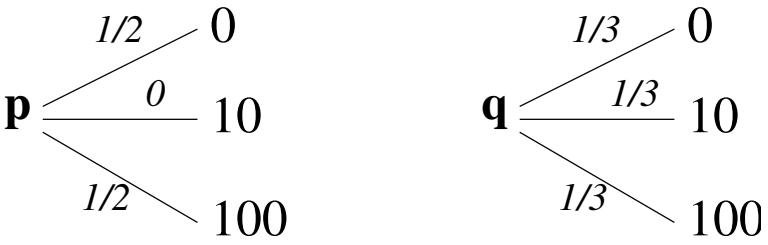


FIG. 1.3 – Exemples de loteries

de loteries sur lesquelles sont données des préférences. Une loterie est constituée d'une distribution de probabilités sur des conséquences. La figure 1.3 présente deux loteries p et q. L'ensemble des conséquences considérées est $X = \{0, 10, 100\}$. D'après la loterie p, la probabilité de la conséquence $x = 0$ est de $0,5$. La théorie de l'utilité espérée affirme que si les préférences entre deux loteries p et q vérifient un certain nombre d'axiomes (complétude, transitivité, continuité, indépendance, monotonie et décomposabilité)[12] alors, il existe une fonction $u : X \to \mathbb{R}$ permettant une représentation numérique des préférences sur les conséquences. Cette fonction, appelée **fonction d'utilité**, est unique à une transformation linéaire positive près. Si on souhaite définir des préférences sur les loteries, une mesure d'**utilité espérée** est utilisée. Elle est obtenue en combinant les probabilités des conséquences et leur utilité :

$$UE(p) = \sum_{x \in X} p(x)u(x)$$

Dans le cadre de l'étude du comportement d'un agent, une conséquence correspond à un état. Chaque loterie représente les différents résultats possibles d'une action en environnement stochastique. L'utilité définit donc des préférences entre les états et l'utilité espérée des préférences entre les actions.

Le **principe de maximisation de l'utilité espérée** affirme qu'un agent rationnel devrait choisir, à tout moment, d'exécuter l'action maximisant son utilité espérée. Développer un processus de prise de décision basé sur le principe de maximisation de l'utilité espérée (MUE) permet donc de garantir un comportement rationnel. Si nous sommes en mesure de calculer de manière exacte l'utilité espérée de chaque comportement possible, nous pourrons déterminer un maximum et en déduire le comportement rationnel optimisant de l'agent. Si la complexité des calculs, nous empêche de déterminer un tel maximum, nous nous dirigerons vers la détermination d'un comportement à rationalité limitée. Notons toutefois que l'utilisation de ce principe n'est pas le seul moyen de prendre une décision rationnelle.

Nous avons défini précédemment un agent rationnel comme un agent maximisant sa mesure

[12] Cette axiomatique a fait l'objet de multiples formulations. Dans sa version originale, Von Neummann et Morgenstern énoncent uniquement les axiomes d'ordre (complétude et transitivité), d'indépendance et de continuité.

de performance. Il est alors naturel de s'interroger sur la relation entre utilité espérée et mesure de performance. Russell et Norvig donnent une réponse dans [Russell et Norvig, 2003] : « *If an agent maximizes a utility function that correctly reflects the performance measure by which its behavior is being judged, then it will achieve the highest possible performance score if we average over the environnements in which the agent could be placed* » [13]. Une mesure de performance définit donc un critère global de rationalité, externe à la manière dont le comportement de l'agent sera calculé. L'utilité espérée quant à elle décrit une mesure locale de la rationalité et permet de définir un critère (MUE) de sélection de l'action rationnelle.

Nous venons d'exposer différents degrés de délibération possibles pour un agent. Notons que cette énumération n'est pas exhaustive, elle recense les processus de décision les plus répandus. Tous les comportements qui viennent d'être décrits supposent cependant que l'environnement de l'agent soit modélisable. Si tel n'est pas le cas, il est possible d'avoir recours à des méthodes d'apprentissage permettant à l'agent d'agir dans un environnement inconnu et d'améliorer ses performances au fur et à mesure de l'acquisition de connaissances.

1.1.4 Notre cadre de travail

Nous venons d'étudier quelques unes des nombreuses définitions de la notion d'agent. Nous avons également présenté les principaux concepts indissociables de cette notion. Nous avons pu remarquer que la définition d'une même notion pouvait différer d'un auteur à un autre. Il nous semble donc essentiel de donner notre définition du concept d'agent, afin de situer précisément notre cadre de travail. Nous allons également énumérer les propriétés des agents que nous traiterons par la suite.

Notre définition du concept d'agent est très proche de celle donnée par Wooldridge et Jennings [Wooldridge et Jennings, 1995].

Définition 6 ***Notre définition d'un agent***

Un agent est une entité située dans un environnement, capable de percevoir cet environnement et d'y agir de manière autonome et rationnelle afin d'atteindre ses objectifs.

Cette définition présente également de nombreuses similarités avec la définition d'un agent rationnel de Russell et Norvig [Russell et Norvig, 2003]. A la différence de Ferber, notre définition n'aborde pas la dimension sociale de l'agent.

[13] Si un agent maximise une fonction d'utilité qui reflète correctement la mesure de performance par laquelle son comportement est évalué, alors il obtiendra les meilleurs performances possibles en réalisant une moyenne sur les environnements dans lesquels l'agent peut évoluer.

L'autonomie caractérisera la prise de décision et aura une définition identique à celle de Wooldridge et Jennings. Ainsi, un agent autonome devra être en mesure de prendre ses décisions sans l'intervention d'un tiers.

En ce qui concerne l'environnement des agents, nous allons, pour le moment, rester dans un cadre très général. Nous ne définirons donc pas, dans ce chapitre, de propriétés précises des environnements considérés.

Jusqu'ici nous avons délibérément mis de côté la notion d'interaction entre agents, bien qu'elle soit un point fondamental dans la définition du concept d'agent donnée par Ferber. Nous avons décrit un agent comme une entité individuelle située dans environnement, sans évoquer l'idée que plusieurs agents pourraient interagir dans un même environnement. Nous allons à présent aborder cette dimension sociale.

1.2 Des agents aux systèmes multi-agents

L'idée d'un agent isolé dans un environnement peut paraître assez restrictive. L'observation du monde qui nous entoure permet de constater que tout système est composé d'un ensemble d'entités individuelles interagissant ensemble.

Nous pouvons au quotidien constater, à différentes échelles, la mise en interaction d'entités. A partir de la réunion de ces entités émanent alors des compétences supérieures aux compétences individuelles. Cette idée est à la base de l'**intelligence collective**, terme désignant les capacités cognitives résultant de l'interaction d'un ensemble d'entités. Ce phénomène, observable dans de nombreuse sociétés animales (oiseux migrateurs, fourmis, poissons, loups, ...), a fait l'objet d'études aussi bien en sciences cognitives qu'en sciences sociales, en économie, en philosophie, ... Ce principe s'est plus largement développé en informatique avec l'apparition de l'Internet permettant de rassembler les connaissances et de mettre plus facilement en relation les individus.

A partir de ces constatations, c'est naturellement que les chercheurs ne se sont plus limités au modèle d'un agent situé dans un environnement, mais qu'ils ont considéré un ensemble d'agents interagissant dans un environnement. Un tel système est appelé **système multi-agent**. A partir de ces systèmes, nous pouvons espérer obtenir des comportements plus élaborés que dans un système mono-agent.

1.2.1 Multiplicité des entités et interactions

Les systèmes multi-agents constituent un domaine assez récent de l'informatique. Etudiés depuis 1980, ils n'ont été largement reconnus qu'au milieu des années 90. Jennings et al. [Jennings *et al.*, 1998] expliquent que les systèmes multi-agents ont tout d'abord constitué un sous-domaine

de l'**Intelligence Artificielle Distribuée** (IAD) qui était alors divisée en deux champs : la résolution distribuée de problèmes (RDP) et les systèmes multi-agents (SMA). La résolution distribuée de problèmes s'intéressait à des problèmes pouvant être découpés en modules résolus de manière distribuée. La stratégie d'interaction entre les modules (par exemple l'échange de solutions partielles), faisait partie intégrante du problème. Les systèmes multi-agents, au contraire, s'intéressaient au comportement d'un ensemble d'agents autonomes et devant résoudre un problème donné. Un système multi-agent était alors vu comme un réseau faiblement couplé de solveurs de problèmes, travaillant ensemble pour résoudre des problèmes qui sont au delà de leurs capacités individuelles.

Ces deux champs faisaient déjà apparaître l'idée d'intelligence collective comme le résultat de la réunion d'un ensemble d'entités.

Plus récemment, le terme « système multi-agent » a pris une signification plus générale. Ferber classe ainsi la résolution distribuée de problèmes comme un domaine d'application des systèmes multi-agents. Par ailleurs, G. Weiss [Weiss, 1999] définit l'intelligence artificielle distribuée comme « l'étude, la conception de systèmes multi-agents ». A l'heure actuelle, un système multi-agent désigne tout type de système composé de plusieurs entités (semi-)autonomes [Jennings *et al.*, 1998].

De la même façon qu'il existe un grand nombre de définitions du terme d'agent, plusieurs définitions des systèmes multi-agents ont été données. Toutes ces définitions s'organisent autour des idées de multiplicité des entités et d'interaction, comme le montre la définition de Wooldridge :

Définition 7 ***Agent selon Wooldridge [Wooldridge, 2002]***

« A muliagent system is one that consists of a number of agents, which interact with one another » [14].

1.2.2 Les caractéristiques d'un SMA

Wooldridge ajoute que les agents composant le système ont différentes « **sphères d'influence** », c'est-à-dire qu'ils contrôlent différentes parties de l'environnement. Ces sphères peuvent, dans certains cas, se chevaucher ce qui entraîne des **dépendances** entre les agents. Cette idée de sphère d'influence traduit la **localité** des actions des agents composants le système. La localité peut également intervenir au niveau des perceptions des agents comme le montre la description des caractéristiques d'un système multi-agent par Jennings [Jennings *et al.*, 1998] : « The characteristics of multiagent systems are :

- each agent has incomplete information, or capabilities for solving the problem, thus each agent has limited viewpoint ;
- there is no global system control ;

[14] Un système multi-agent est constitué d'agents interagissant entre eux.

– data are decentralized » [15].

Ces notions de dépendance et de localité sont présentées par le modèle influence-réaction proposé par Ferber et Müller [Ferber et Muller, 1995]. Ce modèle, représenté pour trois agents sur la figure 1.4, définit un système multi-agent comme étant composé d'un ensemble d'agents Ags et d'un environnement E. Chaque agent exerce une influence sur l'environnement E. La combinaison des influences des agents entraîne une réaction de l'environnement qui évolue en conséquence.

1. Ags définit l'ensemble des agents a tels que $a = \langle Percept, P_a, S_a, F_a, A_a, Infl_a \rangle$ où :
   - $Percept$ définit l'ensemble des perceptions possibles de l'agent.
   - P_a est la fonction de perception de l'agent.
   - S_a correspond à l'ensemble des états internes s_a de l'agent. $s_a(t)$ est l'état interne de l'agent à l'instant t.
   - F_a est une fonction décrivant la dynamique interne de l'agent telle que :

$$s_a(t+1) = F_a(s_a(t), P_a(s_t))$$

   - A_a est l'ensemble des influences possibles de l'agent.
   - $Infl_a$ correspond à la fonction de décision de l'agent. Pour chaque état interne, elle dicte quelle influence exercer.
2. L'environnement E est défini par un tuple $\langle \Gamma, S, T \rangle$ tel que :
   - $\Gamma = \bigcup_{a \in Ags} \Gamma_a$ est l'ensemble des influences que peuvent entreprendre les agents.
   - S est l'ensemble des états possibles de l'environnement. $s(t)$ désigne l'état de E à l'instant t.
   - T définit les lois d'évolution de l'environnement :

$$s(t+1) = T(s(t), \Pi_{a \in Ags} Infl_a(s_a(t))$$

     où Π est l'opérateur de combinaison des influences simultanées des agents.

Ce modèle met en évidence les interactions entre les influences des agents, modélisées par l'opérateur Π. La fonction T tient compte de la combinaison des influences simultanées, mais également des lois d'évolution du monde. Chaque agent base sa décision sur son état interne. Il est important de différencier état interne et état du monde, en raison de la localité des perceptions de l'agent. Une influence modifie l'état de E donc les perceptions de l'agent. Le fonctionnement

[15] Les caractéristiques d'un SMA sont :
- chaque agent a des informations incomplètes ou des compétences limitées pour résoudre le problème, de ce fait chaque agent a un point de vue limité ;
- il n'existe pas de contrôle centralisé ;
- les données sont décentralisées.

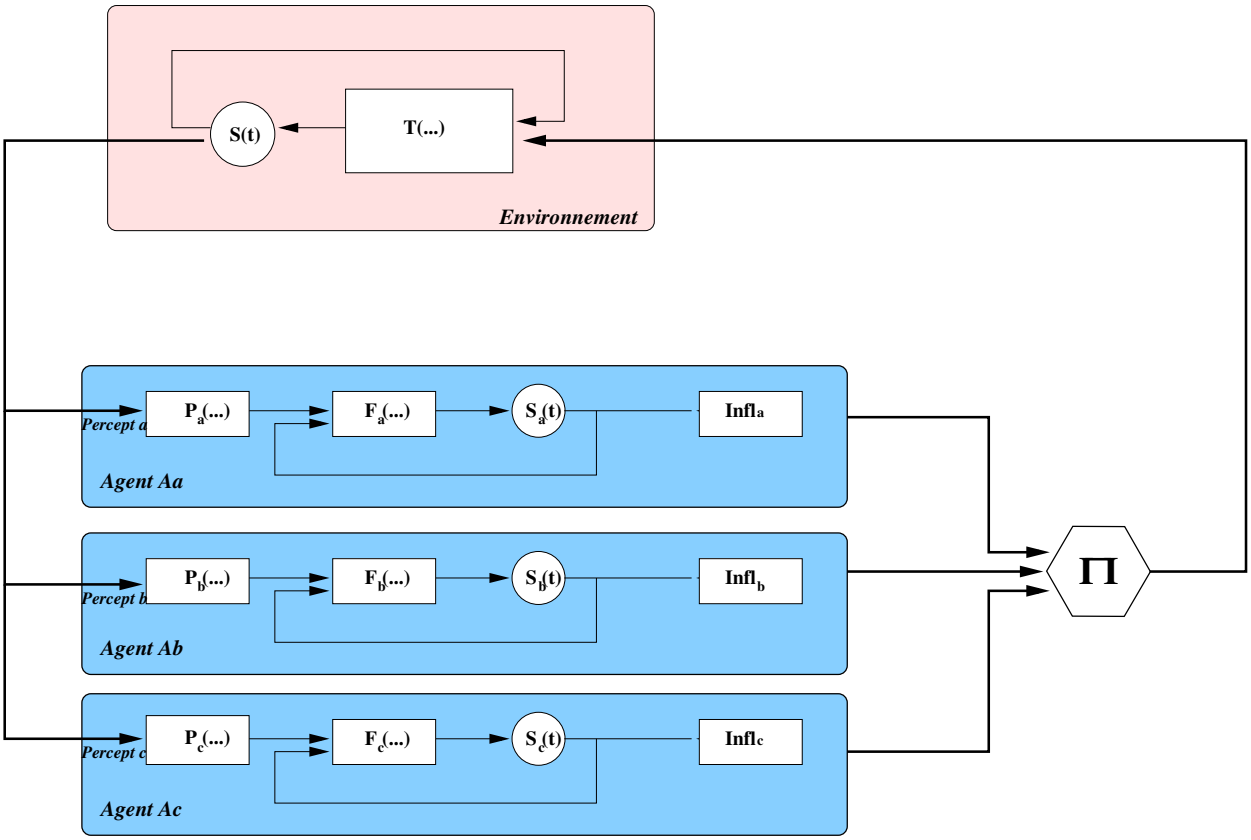


FIG. 1.4 – Modèle influence-réaction pour 3 agents

de chaque agent peut ainsi être assimilé à une boucle perception-action comme celle décrite par la figure 1.1 dans le cas mono-agent. Notons cependant qu'une influence n'est pas une action. En effet, une action consiste, de manière générale, en la modification d'un état. Dans le modèle qui vient d'être présenté, chaque agent exerce une influence. La modification de l'état, ou réaction, résulte de la combinaison des influences des agents et de l'application des lois de l'évolution du monde. Ce n'est donc pas l'action qui modifie l'environnement mais la combinaison des influences des agents.

Pour résumer, nous pouvons définir un système multi-agent comme un ensemble d'agent situés dans un même environnement et interagissant. Les notions d'agent et d'environnement restant identiques à celles introduites précédemment dans le cadre mono-agent, nous allons nous intéresser plus en détail à la troisième notion clef : l'interaction.

1.3 Interactions dans les systèmes multi-agents

Dans les systèmes multi-agents, la multiplicité des entités situées dans un même environnement conduit à des situations d'interaction : toute action d'un agent peut influencer les autres agents.

Ferber [Ferber, 1995] définit l'**interaction** comme « une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques ».

Nous pouvons illustrer ces propos à l'aide d'un exemple. Considérons deux agents (robots) chargés d'explorer ensemble un bâtiment. Supposons qu'à un instant donné, ils doivent tous deux franchir une même porte étroite où ils ne peuvent passer en même temps. Si l'un des agents s'engage dans la porte, il influence l'autre agent en l'empêchant de passer. Le passage de la porte va alors constituer une **situation d'interaction**. Plus formellement, une situation d'interaction est « *un ensemble de comportements résultant du regroupement d'agents qui doivent agir pour satisfaire leurs objectifs en tenant compte des contraintes provenant des ressources plus ou moins limitées dont ils disposent* » [Ferber, 1995].

1.3.1 Les différents types d'interactions

Cette définition permet à Ferber d'établir une classification des types d'interaction selon la nature des buts, les relations aux ressources et les compétences des agents. Trois catégories d'interaction peuvent ainsi être identifiées :

- **indifférence** : les agents sont complètement indépendants, leurs buts sont compatibles, les ressources et les compétences suffisantes. Chaque agent peut donc atteindre ses buts indépendamment des autres agents.
- **coopération** : lorsque les buts des agents sont compatibles mais que les ressources ou les compétences d'un ou plusieurs agents sont insuffisantes, les agents doivent coopérer afin d'atteindre leurs buts.
- **antagonisme** : les buts des agents sont incompatibles. Si les ressources sont suffisantes, les agents se trouvent en situation de compétition. Sinon, la situation d'interaction devient une situation de conflit.

Cette classification peut être affinée en ajoutant d'autres caractéristiques telles que l'existence d'une structure d'autorité ou d'une hiérarchie entre les agents. De plus, les situations peuvent varier suivant le niveau d'interaction considéré. Ainsi, des agents ayant des buts compatibles peuvent localement se trouver en situation de compétition. Nos agents explorant un bâtiment évoluent *a priori* de manière coopérative. Cependant, au passage de la porte, ils peuvent être considérés localement en situation de compétition. En effet, ils souhaitent tous deux accéder à une même ressource : la porte.

La **collaboration** consiste à travailler à plusieurs sur un même projet. Elle désigne les techniques permettant aux agents de se répartir les tâches, les informations et les ressources. Ce terme est à différencier de la **coordination** qui est définie par Malone [Malone, 1988] comme « l'ensemble des activités supplémentaires qu'il est nécessaire d'accomplir dans un environnement multi-agents et qu'un seul agent poursuivant les même buts n'accomplirait pas ». Ces tâches de coordination permettent d'améliorer le comportement des agents. Ainsi, les agents peuvent faire face à des situations où leurs actions sont susceptibles d'interférer entre elles. De même, la coordination permet de garantir une certaine qualité de résultat même si un agent manque de compétences, de ressources ou d'informations.

Selon Ferber [Ferber, 1995] ; « *la coordination des actions, dans le cadre de la coopération, peut donc être définie comme l'articulation des actions individuelles accomplies par chacun des agents de manière à ce que l'ensemble aboutisse à un tout cohérent et performant. (...) l'action du groupe est améliorée soit par une augmentation des performances, soit par une diminution des conflits. La coordination des actions est donc l'une des principales méthodes pour assurer la coopération entre agents autonomes.* ». Collaboration et coordination sont donc deux moyens pour les agents de coopérer.

1.3.2 La communication comme moyen d'interaction

La communication constitue un moyen fondamental de mise en œuvre des méthodes de coopération telles que la coordination ou la collaboration. La communication caractérise tout échange d'information entre des agents. Sans communication, les agents se trouvent isolés les uns des autres et aucune coopération n'est possible.

La théorie de l'information proposée par Shannon et Weaver [Shannon et Weaver, 1948] identifie différentes composantes de l'acte de communication. Ce dernier consiste en une transmission d'informations d'un émetteur vers un récepteur également appelé destinataire. Comme le montre la figure 1.5, l'information est transmise via un canal de transmission pouvant être bruité. Cette information est codée à l'aide d'un langage. Si le code est connu du récepteur, celui-ci pourra décoder l'information et lui donner un sens en s'appuyant sur le contexte du message.

Les types de communication

Différentes catégories de communication ont été répertoriées au cours des nombreuses études sur la théorie de la communication. Les types de communication ont ainsi été classés suivant leur fonction, les éléments de la communication, le type de message, ...

En ce qui nous concerne, nous nous référerons à Goldman et Zilberstein [Goldman et Zilberstein, 2004]. Ces derniers ont répertorié trois types de communication possibles entre agents suivant le mode de transmission du message : la communication directe, la communication indi-

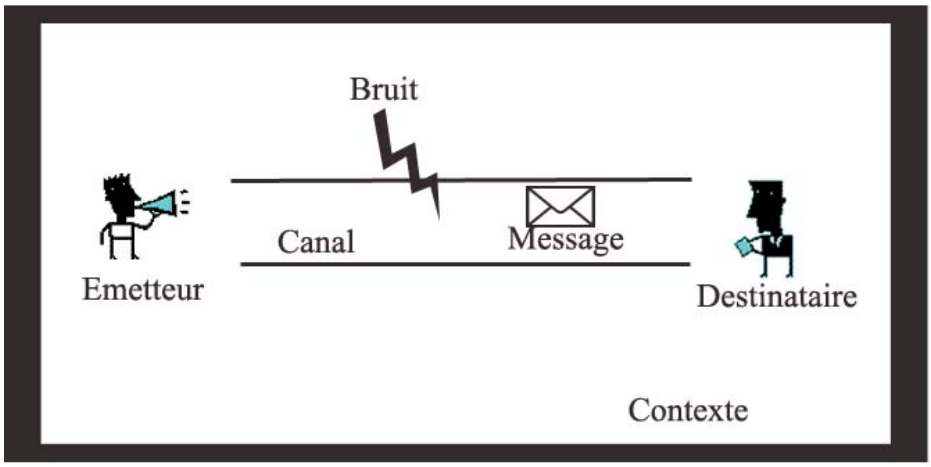


FIG. 1.5 – Composantes de la théorie de l'information

recte, et la communication par observation commune de caractéristiques de l'environnement.

La **communication directe** consiste à envoyer directement un message à un ou plusieurs agents. Lorsque le message est envoyé à un seul agent, on parle de **communication point à point**. Si l'information est envoyée à tous les autres agents, le message est envoyé en **mode diffusion** (ou broadcasting). Dans ce type de communication, le canal de transmission peut être un réseau informatique, un câble téléphonique, des ondes hertziennes, ...

La communication directe peut être **synchrone** ou **asynchrone**. Une communication synchrone est une communication dans laquelle le message est reçu « en temps réel » par le destinataire, c'est le cas d'une conversation téléphonique. Dans une communication asynchrone, l'envoi et la réception du message sont réalisés en temps distincts et séparés par un laps de temps plus ou moins important. L'envoi de courrier par la poste, les forums, le courrier électronique constituent des exemples de communication directe asynchrone. Notons que l'envoi de message par communication directe est généralement intentionnel : l'agent décide d'entrer en interaction avec un ou plusieurs agents et d'envoyer une information précise.

La **communication indirecte** est réalisée par modification de l'environnement ou par manipulation de connaissances communes. Les agents peuvent laisser des traces dans l'environnement afin d'indiquer aux autres quelle action a été exécutée, quel est leur état, etc. Lorsque les agents ont tous accès à une base de données commune, la communication peut s'effectuer par modi-

fication des connaissances stockées dans la base. Dans le cas de la communication indirecte, il n'est pas certain que tous les agents observeront ces modifications. Par conséquent, l'information peut ne pas être reçue. La diffusion d'information par propagation de signal ou par voie d'affiche présentée par Ferber [Ferber, 1995] constitue un mode de communication indirecte. Notons que la communication indirecte est asynchrone. La communication indirecte peut être intentionnelle ou non. En effet, l'agent peut modifier l'environnement sans pour autant vouloir communiquer avec les autres agents.

Le dernier type de communication consiste à **observer des caractéristiques de l'environnement** sur lesquelles les agents n'ont aucune influence. Ces caractéristiques doivent être observables par tous les agents. Suivant les observations réalisées, les agents vont alors décider de la manière de se coordonner, d'agir. Considérons deux agents devant se déplacer dans une même direction, chaque agent ne pouvant pas observer le sens de déplacement de l'autre. Il est alors possible pour les agents de décider de se déplacer vers le nord lorsqu'il pleut et sinon se déplacer vers le sud. Dans ce cas, la caractéristique de l'environnement, observable par les deux agents, est la météo. Grâce à l'observation de l'environnement, une communication implicite s'établit entre les agents. Cependant, ce type de communication suppose qu'un accord préalable soit conclu entre les agents sur le comportement à adopter en fonction des observations. Ce dernier type de communication est bien entendu non-intentionnel. Les agents ne décident en aucun cas de communiquer ou non, ni du contenu du message. Dans ce type de communication, les agents ne sont pas émetteurs du message mais seulement récepteurs.

Les langages de communication

Dans la classification qui précède nous ne nous sommes pas intéressée au problème du codage de l'information. Dans la suite de notre travail, nous n'aborderons pas la question du langage de communication utilisé par les agents. Nous supposerons qu'ils utilisent un même langage connu de tous.

Nous tenons cependant à préciser que l'étude des langages de communication fait l'objet de nombreuses recherches en intelligence artificielle. En ce qui concerne la communication directe, nous pouvons citer les travaux sur la théorie des actes du langage [Austin, 1962, Searle, 1969] qui utilisent le dialogue entre humains comme modèle de communication entre agents. Différents types de messages peuvent ainsi être identifiés ce qui permet de définir des langages structurés d'échange d'information tels que KQML [Finin *et al.*, 1994] ou FIPA-ACL [FIPA, 2006].

En ce qui concerne la communication indirecte, de nombreuses études se sont intéressées aux modes de communication chez les animaux afin de les retranscrire dans un contexte d'agents

informatiques. Nous pouvons citer, à titre d'exemple, les systèmes multi-agents basés sur le principe de communication des fourmis par dépôt de phéromones dans l'environnement [Bonabeau *et al.*, 1999].

1.4 Organisation des interactions

L'interaction implique que les agents aient recours à des méthodes d'allocation de tâches, de spécialisation, de négociation et de communication afin d'augmenter leurs performances. Pour que ces techniques soient efficaces, il est cependant nécessaire qu'une organisation permette de structurer les interactions en définissant les relations entre les entités du système.

Une **organisation** peut être définie par le concepteur du système ou bien émerger comme résultat des interactions entre les agents. Comme l'explique Ferber [Ferber, 1995], ce terme désigne donc « la fois le processus d'élaboration d'une structure et le résultat de ce même processus ». Les organisations pré-établies par un concepteur ne sont pas pour autant obligatoirement statiques. Celui-ci peut en effet concevoir une organisation qui s'adaptera, au cours du temps, à l'évolution du système.

Les différents types d'organisation rencontrés dans les systèmes multi-agents sont généralement inspirés des formes d'organisation humaines sociales ou politiques, ou bien des organisations biologiques. Parmi les modèles d'organisation les plus connus, nous pouvons citer les organisations hiérarchiques [Fox, 1981], prototypes des organisations militaires ou d'entreprise, dans lesquelles les agents sont structurés en niveaux. Les agents d'un niveau donné assurent les fonctions de décision pour le niveau inférieur auquel ils envoient des ordres. Les réseaux contractuels présentés par Smith [Smith, 1980] structurent les interactions sous la forme d'appels d'offres et permettent de procéder à l'allocation de tâches. L'ensemble des agents est composé d'administrateurs gérant les appels d'offres, et d'offrants élaborant des propositions et exécutant les tâches. Il est possible de greffer une structure hiérarchique à ce type d'organisation en élaborant des niveaux d'appels d'offres. Différents niveaux d'administrateurs sont alors créés. Enfin, nous pouvons mentionner les travaux de Shoham et Tennenholtz [Shoham et Tennenholtz, 1992] qui proposent un modèle d'organisation basé sur l'instauration de lois sociales décrivant comment les agents doivent se comporter.

Nous avons présenté les concepts indissociables de tout système multi-agent. Les notions d'agent et d'environnement ayant été décrites précédemment, nous nous sommes plus particulièrement intéressée aux interactions entre agents et à leur organisation.

Ces quatre composantes d'un système multi-agent se retrouvent dans l'approche Voyelle proposée par Demazeau [Demazeau, 1997]. Cette dernière définit un système multi-agent comme l'ensemble des quatre composantes AEIO : Agent, Environnement, Interaction et Organisation.

1.5 Problématique de la prise de décision dans les SMA

Nous allons à présent revenir plus en détail sur certaines propriétés des systèmes multi-agents, liées aux caractéristiques conjuguées des agents et de l'environnement. Nous allons ainsi développer les notions d'observabilité et d'incertitude qui occuperont une place importante dans la suite de notre travail, en particulier pour la prise de décision dans les systèmes multi-agents.

1.5.1 Observabilité

L'**observabilité** d'un environnement caractérise l'ensemble des informations qui sont accessibles à un agent. Il est également possible de parler de perception. Nous avons déjà abordé les idées de localité et d'imprécision des perceptions d'un agent, introduisant ainsi la notion d'observabilité partielle. De manière générale, nous parlerons d'**observabilité partielle** quand toutes les informations nécessaires à la prise d'une décision (ou phase de délibération) ne sont pas accessibles. Les agents seront alors contraints de faire face à ce manque de connaissances et devront décider au mieux comment agir en fonction des informations dont ils disposent.

Suivant les informations auxquelles les agents ont accès, nous pouvons distinguer différents types d'observabilité. Nous différencierons tout d'abord l'observabilité de l'état global du système et l'observabilité de l'état local de l'agent. Le terme état global désignera l'état du système multi-agent (agents + environnement). L'état local d'un agent sera défini selon le degré de délibération de l'agent. Il pourra correspondre aux perceptions courantes de l'agent ou bien à une représentation interne de ses connaissances.

L'état local d'un agent peut être partiellement ou complètement observable. Dans le premier cas, on parle d'état **localement partiellement observable**. L'agent n'a alors pas accès à toutes les informations nécessaires pour savoir dans quel état local il se trouve. Dans le cas contraire, l'état est **localement totalement observable** : l'agent peut, à partir de ses observations, déduire avec certitude son état local.

En ce qui concerne l'état global du système, on distingue **observabilité collective** et **observabilité individuelle**. L'observabilité collective résulte de l'agrégation des observations de tous les agents. L'état global du système peut être **collectivement partiellement observable** ou bien **collectivement totalement observable**. Dans le cas de l'observabilité collective partielle, il n'est pas possible de déduire l'état du système même si on a accès à toutes les observations de tous les agents. Certaines propriétés de l'état global ne sont donc observées par aucun des agents. Dans le cas contraire, on peut déduire l'état global du système à partir des observations locales et l'état global est dit collectivement totalement observable.

Par ailleurs, l'état global du système est dit **individuellement observable** si chaque agent connaît l'état global du système à partir de ses observations. Un état global individuellement observable est par conséquent collectivement totalement observable.

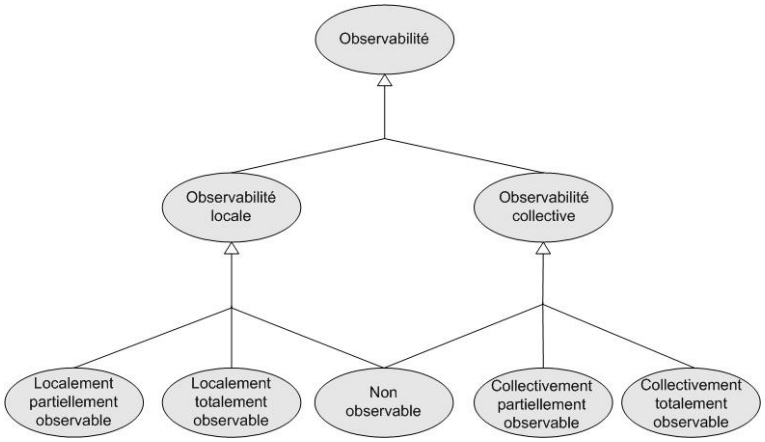


FIG. 1.6 – Différents types d'observations

Enfin, lorsque les agents n'ont aucune observabilité, on parle de **non-observabilité** (locale et collective).

Remarquons que l'observabilité ne concerne pas seulement l'état du système et des autres agents mais également leur comportement. En effet, chaque agent peut, dans certains cas, connaître les stratégies des autres agents. Il est est ainsi possible de déduire comment s'adapter au mieux à leur comportement.

Nous montrerons par la suite que le degré d'observabilité influence la complexité de la prise de décision. Intuitivement, si nous connaissons tout sur les états de l'environnement, les états des autres agents, et leur comportement, il est assez facile de décider comment agir. Si nous ne disposons que d'informations partielles sur les autres agents, nous sommes contraints d'envisager tous leurs comportements possibles, ce qui complexifie la décision.

1.5.2 Incertitude

L'idée d'**incertitude** a également déjà été abordée lorsque nous avons discuté de l'aspect stochastique des environnements et de la précision des capteurs et des effecteurs d'un agent.

Il existe quatre causes d'incertitude dans un système multi-agent :

- l'imprécision et les limitations des capteurs,
- l'imprécision des effecteurs,
- la modélisation incomplète de l'environnement,
- les interactions entres agents.

L'imprécision des capteurs entraîne des incertitudes sur l'information perçue par l'agent et par conséquent sur l'environnement. En effet, les capteurs peuvent être bruités ou avoir des capacités limitées, ce qui induit des erreurs dans les observations réalisées. De plus, chaque capteur est restreint par une portée (ou « champ de vision ») qui ne lui permet généralement pas de percevoir

tout l'environnement. Tout ce qui est hors de portée des capteurs se trouve alors non-observable et par conséquent incertain.

L'imprécision des effecteurs ainsi que l'incomplétude de la modélisation de l'environnement, entraînent par ailleurs des incertitudes sur le résultat des actions des agents. L'incomplétude de la modélisation est essentiellement due au concepteur pour lequel il est, de manière générale, impossible de modéliser parfaitement l'environnement dans lequel les agents évoluent. Le concepteur peut, en effet, avoir une connaissance incomplète de tous les phénomènes et lois de l'environnement. De plus, même si le concepteur a accès à toutes ces connaissances, il est généralement impossible de rendre compte de tous les paramètres intervenant dans l'évolution du système. Il est par exemple difficile de modéliser l'ensemble des forces (frottements, glissements, facteurs météorologiques, ...) régissant les déplacements d'un robot. Il peut par ailleurs être nécessaire de procéder à une discrétisation de l'environnement entraînant alors une perte de précision des données modélisées. De cette incomplétude de la modélisation découlent des incertitudes sur les effets des actions des agents. Les environnements dans lesquels évoluent les agents sont alors considérés comme stochastiques et les incertitudes sont représentées à l'aide de distributions de probabilités.

Enfin, chaque agent n'a généralement qu'une observabilité partielle des états des autres agents et de leur comportement. Cependant, le résultat de l'exécution d'une action dépend, en partie, des actions entreprises par les autres agents. En effet, les agents évoluent dans un même environnement, leurs actions peuvent donc entrer en interaction, conduisant alors à des effets non souhaités ou non envisagés. Nous avons déjà abordé cette idée lors de la description du modèle influence-réaction proposé par Ferber et Müller. Un opérateur Π de combinaison des influences simultanées avait alors été introduit afin de rendre compte des interactions entre les influences des agents. Le fait qu'un agent ne connaisse pas les actions entreprises par les autres agents ajoute donc un degré d'incertitude sur le résultat de ses actions.

Pour qu'un agent évolue dans un environnement déterministe, il serait nécessaire que le système soit parfaitement modélisé, que les capteurs et effecteurs des agents soient d'une précision irréprochable, que la portée des capteurs soit illimitée et que l'agent connaisse exactement les états et comportements des autres agents. Il paraît évident qu'un tel système est difficilement envisageable.

1.5.3 Délibération et décision

Dans le cadre de cette thèse, nous nous intéresserons au processus de délibération des agents et plus particulièrement à la prise de décision. Nous avons choisi de revenir plus en détail sur l'observabilité et l'incertitude car elles sont à l'origine de nombreuses difficultés posées lors de la prise de décision dans les systèmes multi-agents.

Nous considérerons par la suite des systèmes multi-agents coopératifs. Nous chercherons alors à utiliser la multiplicité des entités afin d'augmenter les performances du système. Pour ce faire, des méthodes assurant la coopération entre agents devront être mises en œuvre afin de tirer parti, au mieux, de la pluralité des agents.

Lorsque l'observabilité est partielle et le résultat des actions incertain, il est d'autant plus difficile pour les agents de se coordonner. Il est possible de contourner le problème en faisant appel à une entité centrale prenant toutes les décisions et coordonnant les actions de chacun (un chef de projet). Dans les applications que nous considérerons, nous montrerons qu'une telle solution est cependant impraticable. Chaque agent devra donc décider de manière autonome comment agir dans un environnement incertain (stochastique) tout en faisant face au manque d'observabilité. Dans le chapitre suivant, nous proposerons un formalisme mathématique pour la résolution de problème décisionnel mono-agent sous incertitude. Celui-ci sera étendu au cas multi-agent dans le chapitre 3, nous reviendrons alors plus en détail sur les problèmes de coordination et d'observabilité partielle.

1.6 Domaines d'application des SMA

Avant de conclure ce chapitre, nous tenons à réaliser une ébauche de la diversité des applications des systèmes multi-agents. L'intérêt pour les systèmes à base d'agents a connu une croissance importante ces dernières années. Ainsi, les applications des systèmes multi-agents se sont largement diversifiées : pilotage automatique d'avions, recherche d'informations, e-learning, ...

Woooldridge [Wooldridge, 2002] détaille un certain nombre d'applications et propose une méthodologie pour la mise en œuvre de systèmes à base d'agents. Parmi les nombreuses applications des systèmes multi-agents, nous pouvons citer la simulation et l'étude des phénomènes biologiques ou économiques. Des environnements virtuels peuvent ainsi être créés afin d'étudier le comportement d'individus (évolution d'un gène dans une population, transmission de virus, échanges commerciaux, etc.).

Diverses applications liées à l'Internet font également appel à de tels systèmes : recherche d'information, indexation de données, aide à la navigation, enchères et commerce électroniques, ... De manière générale, les systèmes multi-agents sont utilisés dans des systèmes distribués où il est nécessaire de gérer la décentralisation des données, des connaissances et du contrôle.

Pour finir, nous tenons à mettre en avant l'importance des systèmes à base d'agents en robotique. En effet, les robots constituent naturellement des agents physiques équipés de capteurs et d'effecteurs leur permettant de percevoir leur environnement et d'y agir.

Conclusion

Dans ce chapitre nous avons détaillé une composante essentielle de l'intelligence artificielle : la notion d'agent. Nous nous sommes tout d'abord intéressée à des systèmes mono-agent constitués d'un agent et d'un environnement. Malgré la diversité des définitions recensées, nous avons réussi à identifier les caractéristiques d'un agent et à établir notre propre définition du terme d'agent.

Nous avons ensuite mis en avant l'aspect limitatif du point de vue mono-agent et nous avons introduit les systèmes multi-agents. Nous avons alors développé les idées de multiplicité des entités, d'interaction et d'organisation. Enfin, nous sommes revenue plus en détail sur les notions d'observabilité partielle et d'incertitude. Nous avons ainsi exposé certaines difficultés auxquelles peuvent être confrontés les agents lors de la délibération.

Chapitre 2

Les Processus Décisionnels de Markov

Dans le chapitre précédent, nous avons exposé les difficultés rencontrées pour modéliser de façon parfaite un système réel. Afin de rendre compte de l'imprécision des modélisations, nous avons introduit la notion d'incertitude. Ainsi, des distributions de probabilités sont utilisées afin de représenter l'incertitude émanant d'une modélisation imprécise de l'environnement et des lois le gouvernant. Nous avons également montré que l'incertitude pouvait provenir d'un manque d'observabilité de l'environnement et d'une méconnaissance du résultat des interactions avec les autres agents.

Dans ce chapitre, nous allons décrire différents modèles stochastiques permettant de représenter et d'étudier des systèmes régis par des lois probabilistes. Un système stochastique est constitué d'un ensemble d'états représentant les situations dans lesquelles le système peut se trouver. Le passage d'un état à un autre est décrit par une fonction de probabilités appelée fonction de transition.

Nous commencerons par présenter une propriété majeure de certains modèles stochastiques : la propriété de Markov. Nous introduirons ensuite les chaînes de Markov qui constituent la base des modèles Markoviens. Nous montrerons qu'elles proposent un formalisme pour la description de systèmes stochastiques mais qu'elles ne s'avèrent pas adaptées à la représentation de systèmes dont l'évolution est influencée par un décideur. Nous décririons alors une extension des chaînes de Markov, les Processus Décisionnels de Markov (MDP), et nous détaillerons deux principaux algorithmes dédiés à leur résolution : Value Iteration et Policy Iteration. Enfin, nous discuterons des champs d'application des MDPs pour la résolution de problèmes décisionnels sous incertitude.

2.1 La propriété de Markov

Les systèmes stochastiques ont été largement étudiés au début du $XX^{\text{ème}}$ siècle par le mathématicien russe Andrei Andreyevich Markov (1856-1922) qui s'est plus particulièrement intéressé aux systèmes possédant la **propriété de Markov**. Cette dernière affirme que la connaissance de l'état d'un système à l'instant t est suffisante pour prédire l'évolution du système à l'instant $t+1$. Il n'est donc pas nécessaire de connaître l'historique des états passés, l'état courant suffit. Plus formellement, nous pouvons résumer cette propriété par l'équation suivante :

$$P(s_{t+1} = s' | s_0, s_1, \cdots, s_t) = P(s_{t+1} = s' | s_t)$$

Cette propriété peut paraître très restrictive. De nombreux systèmes à première vue non markoviens peuvent cependant être modélisés de façon à vérifier la propriété de Markov. En effet, tout dépend de la manière dont sont définis les états du système. Si ces derniers sont construits de façon à contenir toutes les informations nécessaires à la prédiction de l'évolution du système, le modèle est markovien.

Prenons l'exemple d'un train pouvant se déplacer entre Caen et Paris. Soit il part de Caen et va vers Paris, soit il fait le trajet en sens inverse. Supposons que chaque état du système corresponde à la dernière gare dans laquelle le train s'est arrêté. L'information contenue dans l'état courant est alors insuffisante pour prédire l'état suivant. En effet, il est également nécessaire de connaître la direction du train. Si les deux dernières gares, c'est-à-dire les deux derniers états, sont connues il est possible de déduire le sens du train. Le système n'est toutefois pas markovien puisqu'il est nécessaire de connaître l'état courant et l'état précédent afin de prédire l'état suivant. En revanche, si un état est constitué du nom de la dernière gare et du sens de déplacement du train ou bien des deux dernières gares alors, nous sommes en mesure de prévoir le prochain arrêt à partir de l'état courant, le système est markovien.

Un état n'est donc pas restreint aux perceptions disponibles à un instant donné. Il peut être construit petit à petit et contenir un historique des évènements passés. Dans ce cas, un état peut résumer toutes les informations nécessaires à la prévision de l'évolution du système. De ce fait, de nombreux systèmes stochastiques peuvent être décrits comme des systèmes markoviens. Nous pouvons cependant attirer l'attention sur le fait que plus la quantité d'information devant être contenue dans un état est importante, plus l'espace d'états est grand. Il est donc important que les états ne contiennent que l'information utile à la décision. Dans notre précédent exemple, il est inutile que les états du système soient constitués du nombre de passagers présents dans le train. Cette donnée n'a pas d'impact sur la prochaine gare par laquelle passera le train. De plus, la prise en compte de cette donnée entraînerait une explosion de l'espace d'états en raison de la diversité du nombre de passagers possibles.

2.2 Les chaînes de Markov

Les **chaînes de Markov** permettent de représenter la dynamique des systèmes dont l'évolution est régie par des lois stochastiques vérifiant la propriété de Markov. Elles sont utilisées dans de nombreux domaines allant de l'économie à la génétique comme par exemple la gestion d'entreprises, de clientèle, l'étude d'automates et de langages, la représentation de phénomènes d'attente, de processus d'usure ou la recherche d'associations de gènes.

2.2.1 Quelques définitions

Une chaîne de Markov est constituée d'un ensemble d'états $\mathcal{S}$ et d'une fonction de transition $\mathcal{P}$ indiquant la probabilité $\mathcal{P}(s'|s)$ de passer d'un état s à un état s'. Nous limiterons notre description aux chaînes de Markov à espace d'états finis. Il est alors possible de représenter ces données par un graphe ou bien par une matrice de transitions. De ce fait, l'étude des chaînes de Markov à espaces d'états finis est étroitement liée à la théorie des graphes orientés ainsi qu'au calcul matriciel.

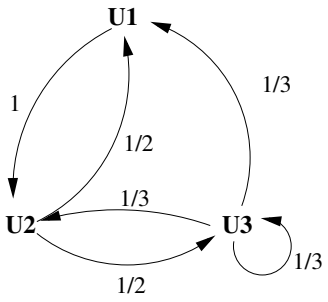
Dans une représentation sous forme de graphe, les nœuds du graphe correspondent aux états du système et les arêtes aux transitions entre les états. Chaque arête allant d'un état s à un état s' est étiquetée par une probabilité correspondant à la valeur de $P(s'|s)$.

Considérons trois urnes numérotées de 1 à 3. Chaque urne contient des boules également numérotées de 1 à 3. Les boules sont réparties de la façon suivante :

- L'urne 1 U_1 ne contient que des boules numérotées 2.
- L'urne 2 U_2 contient des boules numérotées 1 et 3 en proportions égales.
- L'urne 3 U_3 contient des boules numérotées 1, 2 et 3 en proportions égales.

Un joueur commence par tirer une boule dans une urne choisie arbitrairement. Suivant le chiffre i de la boule, le joueur passe à l'urne U_i et tire à nouveau une boule. On suppose que les boules sont replacées dans leur urne à chaque tirage. Ce jeu peut être représenté par une chaîne de Markov à 3 états dans laquelle chaque état correspond au numéro de l'urne. La figure 2.1 donne la représentation de la chaîne sous forme de graphe et de matrice de transitions. Notons que si les boules ne sont pas replacées dans les urnes après chaque tirage, le modèle n'est plus markovien à moins que la composition de chaque urne soit une donnée de l'état. Dans ce cas, la définition d'un état est modifiée et l'espace des états est beaucoup plus important.

Dans cet exemple, le passage d'un état s à un état s' n'est pas obligatoirement direct et peut nécessiter plusieurs transitions. Un états s' est accessible à partir d'un état s, s'il existe un chemin entre s et s' dans le graphe représentant la chaîne de Markov. Calculer la probabilité de passer d'un état s à un état s' revient à calculer la somme des probabilités des chemins allant de s à s'. Calculons la probabilité de passer de l'urne U_1 à l'urne U_3 en quatre tirages exactement.



$$M = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

FIG. 2.1 – Chaîne de Markov pour le problème des urnes

La figure 2.2 présente les différents chemins possibles. La probabilité $P_{1,3}$ est alors calculée de la manière suivante :

$$P_{1,3} = P(U_2|U_1) * P(U_3|U_2) * P(U_2|U_3) * P(U_3|U_2) + P(U_2|U_1) * P(U_3|U_2) * P(U_3|U_3) * P(U_3|U_3)$$

$$+P(U_2|U_1) * P(U_1|U_2) * P(U_2|U_1) * P(U_3|U_2) = 1/12 + 1/18 + 1/4 = 14/36 = 7/18$$

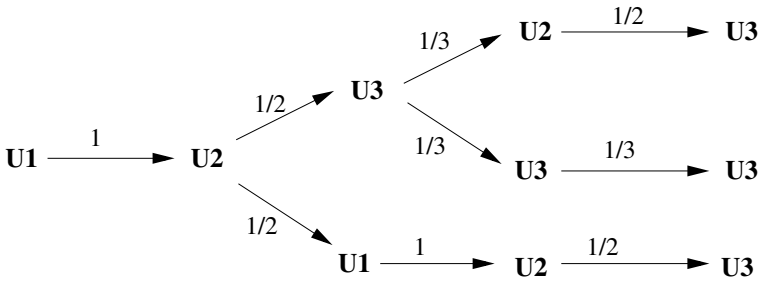


FIG. 2.2 – Chemins entre U_1 et U_3

L'analyse d'une chaîne de Markov permet d'établir des prédictions sur l'évolution du système. Ceci revient à calculer la probabilité de certains états futurs. Les chaînes de Markov permettent également de calculer la fréquence entre les passages par un même état. Enfin, il est possible d'analyser la topologie des graphes les représentant afin de déceler des états ou ensembles d'états absorbants. Un ensemble d'états absorbants constitue un ensemble d'états du système dont il n'est pas possible de sortir. Des transitions entre les états de cet ensemble sont possibles mais, il n'existe pas de transition permettant d'aller vers un état n'appartenant pas à l'ensemble.

2.2.2 Chaînes de Markov cachées

Nous avons jusqu'à présent supposé que les états du système étaient totalement observables. Cette hypothèse n'est pas toujours vérifiée. Dans certains cas, l'état du système peut être inconnu ou partiellement connu. Dans notre exemple, le joueur observe le chiffre de la boule qui a été

tirée, il lui est donc possible de savoir dans quelle urne il tirera la prochaine boule. Si le joueur n'observe pas le numéro de la boule, l'état du système est inconnu. Notons que s'il connaît la répartition des boules dans les urnes, il peut toutefois calculer des probabilités sur les états possibles.

De tels systèmes partiellement observables peuvent être modélisés par des **chaînes de Markov cachées** ou **Hidden Markov Models** (HMM). Le modèle est alors composé d'un ensemble d'états $\mathcal{S}$, d'un ensemble d'observations Ω, d'une fonction de transitions $\mathcal{P}$ et d'une fonction d'observation $\mathcal{O}$. Cette dernière associe à chaque état s et à chaque observation o, une probabilité d'observation : $\mathcal{O}(o, s)$ est la probabilité d'observer o sachant qu'on est dans l'état s. En utilisant la loi de Bayes[16], il est alors possible de déduire la probabilité d'être dans un état s sachant qu'on a observé o et que nous étions auparavant dans l'état s'. La probabilité d'une suite d'observations peut également être calculée.

Les chaînes de Markov cachées sont par exemple utilisées en reconnaissance de la parole [Rabiner et Juang, 1993]. Elles permettent de modéliser une suite de phonèmes sans savoir pour autant à quels mots ils correspondent. Des applications des chaînes de Markov sont également répertoriables en robotique mobile afin de résoudre des problèmes de localisation [Thrun *et al.*, 1998].

2.3 Les Processus décisionnels de Markov

Les chaînes de Markov permettent de décrire la dynamique de systèmes stochastiques dans lesquels l'observateur n'a aucun moyen d'influencer l'évolution du système. Nous qualifierons cet observateur de « passif », il observe le système mais n'intervient pas sur celui-ci. Dans notre exemple, le joueur passe d'une urne à une autre en suivant des règles pré-établies. Il n'a aucun moyen de décider par lui même dans quelle urne tirer la prochaine boule.

Les chaînes de Markov s'avèrent donc peu adaptées pour la représentation de problèmes dans lesquels un agent peut intervenir sur le système. L'agent est dans ce cas actif et peut choisir d'exécuter une action plutôt qu'une autre afin d'influencer l'évolution du système.

Les processus décisionnels de Markov permettent de formaliser un tel système. Ils constituent une extension des chaînes de Markov dans lesquelles sont ajoutés un ensemble d'actions et une fonction de récompense.

[16] $P(A.B) = P(A|B) \times P(B) = P(B|A) \times P(A)$

2.3.1 Les composantes d'un MDP

Définition 8 ***Processus Décisionnels de Markov (MDP)***

Un Processus Décisionnel de Markov (MDP) est défini par un tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ *tel que :*

- $\mathcal{S}$ *est un ensemble fini d'états* s*.*
- $\mathcal{A}$ *est un ensemble fini d'actions* a*.*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ *est une fonction de transition markovienne donnant la probabilité de passer de l'état* s *à l'état* s' *quand l'action* a *est exécutée.*
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ *est une fonction de récompense.* $\mathcal{R}(s, a)$ *est la récompense obtenue par l'agent lorsqu'il exécute l'action* a *à partir de l'état* s*.*

Les processus décisionnels de Markov modélisent des problèmes de décision en environnement stochastique. Dans chaque état, l'agent doit décider quelle action exécuter, il peut ainsi agir sur l'évolution du système.

L'ensemble des actions

L'exécution d'une action a à partir d'un état s est stochastique et peut donc mener à différents états s'. Cet aspect probabiliste est représenté par la fonction de transition.

Le formalisme des MDPs suppose que toute action ait une durée d'une unité de temps. A chaque instant t, l'agent exécute une action qui se termine à l'instant $t + 1$. A cet instant, l'agent doit alors prendre une nouvelle décision. Lorsque les actions ont toutes la même durée, l'étape de décision t (ou $t^{\text{ème}}$ décision) et l'instant t coïncident. Les **processus décisionnels de Markov Semi-Markoviens** s'intéressent quant à eux à la modélisation de problèmes dans lesquels les actions ont différentes durées possibles [Sutton *et al.*, 1999, Puterman, 2005]. Dans de tels systèmes, étape de décision et instant doivent être différenciés.

L'**horizon** d'un MDP définit une limite temporelle au delà de laquelle ce qui peut arriver ne nous intéresse pas. Dans un MDP à horizon fini $\mathcal{H}$, les performances de l'agent sont évaluées en considérant uniquement ses $\mathcal{H}$ prochaines décisions.

La fonction de transition

La fonction de transition d'un MDP formalise la dynamique du système. Elle vérifie la **propriété de Markov**. Étant donné que l'action exécutée par l'agent influence l'évolution du système, la propriété de Markov doit tenir compte des actions. Elle se résume alors par l'équation suivante :

$$P(s_{t+1} = s' | s_0, a_0, s_1, a_1, \cdots, s_t, a_t) = P(s_{t+1} = s' | s_t, a_t)$$

Comme pour les chaînes de Markov, chaque état doit être défini de façon à résumer toute l'information nécessaire à la prédiction de l'évolution du système, et par conséquent à la décision.

Lorsque la dynamique du système est indépendante du temps, le système est dit **stationnaire**. La probabilité de passer d'un état s à un instant t à un état s' à un instant $t+1$ est donc indépendante de t, d'où :

$$P(s'_{t+1}|s_t, a_t) = P(s'|s, a) \quad \forall t$$

Par la suite, nous restreindrons nos propos aux processus décisionnels de Markov stationnaires.

La fonction de récompense

Le fonction de récompense d'un MDP permet de représenter les objectifs de l'agent et de guider son comportement. Elle définit les situations préférées ainsi que les comportements non désirés de l'agent. Elle associe à chaque couple état - action (s, a), une valeur numérique traduisant le fait que l'exécution de l'action a soit souhaitée ou non à partir de l'état s. Suivant la valeur associée à ce couple, l'agent favorisera ou évitera l'exécution de l'action a à partir de s.

Il existe d'autres façons de définir une fonction de récompense. Il est possible de récompenser le fait de passer d'un état s à un état s' ($\mathcal{R}(s, s')$) ou bien tout simplement le fait d'être dans un état s' ($\mathcal{R}(s')$).

A chaque décision, l'agent reçoit donc un signal destiné à le récompenser ou le pénaliser. Le comportement de l'agent dépend de la façon dont il interprète ce signal. Généralement, l'agent cherche à maximiser la somme de ses récompenses. Soit r_t la récompense obtenue à l'étape t. Pour des problèmes à horizon fini $\mathcal{H}$, à chaque étape t, l'agent cherche à maximiser ses $\mathcal{H}$ récompenses à venir, soit :

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_{t+\mathcal{H}} \tag{2.1}$$

Lorsque l'horizon du problème est infini, un facteur d'atténuation $\gamma \in [0, 1[$ est utilisé afin de permettre à l'agent de calculer une récompense pondérée à long terme. On obtient alors :

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \tag{2.2}$$

La valeur du facteur d'atténuation influence le comportement de l'agent. Lorsque γ est proche de 0, les récompenses obtenues dans un futur lointain seront considérées comme insignifiantes et l'agent favorisera ses récompenses à court terme. Si γ tend vers 1, l'agent considérera indifféremment ses récompenses à court terme et à long terme.

2.3.2 L'observabilité partielle dans les MDPs

Comme les chaînes de Markov, les processus décisionnels de Markov supposent que l'agent connaisse l'état dans lequel il se trouve. Nous avons montré, au chapitre précédent, que les agents n'avaient pas toujours une observabilité totale de leur état. De la même manière que les

chaînes de Markov cachées constituent une extension des chaînes de Markov pour des environnements partiellement observables, les processus décisionnels de Markov partiellement observables (POMDP) [Astrom, 1965, Drake, 1962, Smallwood et Sondik, 1973] étendent les MDPs. Un ensemble d'observations ainsi qu'une fonction d'observation sont ainsi ajoutés à la définition d'un MDP.

Définition 9 ***Processus Décisionnels de Markov Partiellement Observable (POMDP)***

Un Processus Décisionnel de Markov Partiellement Observable (POMDP) est défini par un tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \Omega, \mathcal{O}, \mathcal{R} >$ *tel que :*

- $\mathcal{S}$ *est un ensemble fini d'états* s*.*
- $\mathcal{A}$ *est un ensemble fini d'actions* a*.*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ *est une fonction de transition markovienne donnant la probabilité de passer de l'état* s *à l'état* s' *quand l'action* a *est exécutée.*
- Ω *est l'ensemble fini des observations* o *de l'agent.*
- $\mathcal{O} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow \mathbb{R}$ *définit la fonction d'observation.* $\mathcal{O}(s, a, s', o)$ *correspond à la probabilité que l'agent observe* o *lorsque l'action* a *est exécutée à partir de l'état* s *et que le système arrive dans l'état* s'*.*
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ *est une fonction de récompense.* $\mathcal{R}(s, a)$ *est la récompense obtenue par l'agent lorsqu'il exécute l'action* a *à partir de l'état* s*.*

Les relations entre chaînes de Markov, chaînes de Markov cachées, MDP et POMDP sont résumées par le tableau 2.1.

	Observabilité totale	Observabilité partielle
Agent passif	Chaînes de Markov	Chaînes de Markov cachées
Agent décisionnel	MDP	POMDP

TAB. 2.1 – Relations entre les différents modèles markoviens

Le passage d'un modèle à un autre dépend du degré d'observabilité de l'état du système ainsi que de la finalité du modèle. S'il s'agit de se placer en tant qu'observateur de la dynamique du système (agent passif), les chaînes de Markov sont utilisées. S'il est possible d'influencer l'évolution du système par l'exécution d'actions (agent décisionnel), le formalisme des processus décisionnels de Markov est employé.

Dans le cadre de cette thèse, nous allons nous intéresser à des problèmes décisionnels dans lesquels des agents doivent déterminer comment agir afin de maximiser leurs récompenses. Par conséquent, nous étudierons plus particulièrement le formalisme des processus décisionnels de Markov.

2.4 L'optimalité dans les MDPs

Nous avons présenté le formalisme des processus décisionnels de Markov. Ce dernier permet de modéliser des problèmes décisionnels en environnement stochastique. $\mathcal{S}$ et $\mathcal{P}$ décrivent le système et son évolution. $\mathcal{A}$ fournit l'ensemble des actions possibles, et la fonction de récompense $\mathcal{R}$ permet de définir les objectifs de l'agent. Nous allons maintenant nous intéresser à la résolution d'un tel problème. Résoudre un MDP consiste à déterminer un comportement de l'agent qui lui permette d'atteindre ses objectifs. Pour ce faire, il est nécessaire de déterminer une politique d'action de l'agent.

2.4.1 La notion de politique

Une politique est une fonction décrivant comment l'agent doit agir dans chaque situation possible. La politique π d'un agent fait correspondre à chaque état s du système, une action a que l'agent doit exécuter, d'où :

$$\pi : \mathcal{S} \rightarrow \mathcal{A}$$

Le système étant markovien, seul l'état courant est nécessaire pour décider quelle action réaliser. Notons que l'application d'une politique peut être représentée par une chaîne de Markov. La probabilité de passer d'un état s à un état s' est alors déterminée à partir de la politique π et de la fonction de transition du MDP.

Dans un POMDP, l'agent n'a pas accès à l'état du système, la politique est alors une fonction de l'ensemble d'observation Ω vers l'ensemble d'actions $\mathcal{A}$, soit :

$$\pi : \Omega \rightarrow \mathcal{A}$$

Par la suite, nous limiterons nos propos aux processus décisionnels de Markov totalement observables (MDP). Plus de détails concernant les POMDPs peuvent être consultés dans [Murphy, 2000, Kaelbling *et al.*, 1998].

Pour chaque problème de décision formalisé sous forme de MDP, il existe un ensemble Π de politiques possibles $\pi \in \Pi$. Résoudre un MDP consiste à trouver la meilleure politique, c'est-à-dire celle maximisant le critère de performance. Cette solution est appelée politique optimale et est notée π^*.

Bellman [Bellman, 1957] a démontré que tout MDP possédait au moins une politique optimale déterministe et stationnaire. De ce fait, la recherche de la politique peut être réalisée dans l'espace des politiques déterministes[17].

[17] Une politique déterministe associe à chaque état une et une seule action. Elle s'oppose aux politiques stochastiques (ou stratégies mixtes) associant à chaque état différentes actions possibles suivant une certaine distribution de probabilités.

2.4.2 L'évaluation d'une politique

Afin d'établir un classement des politiques et de pouvoir déterminer la meilleure d'entre elles, un critère de performance doit être spécifié. Il permet de définir une mesure d'utilité associant à chaque politique π, une valeur V^{π}. En fonction de leurs valeurs, les politiques pourront alors être classées et la politique optimale sera identifiée. Nous avons précédemment montré (équations 2.1 et 2.2) que le critère de performance d'un agent rationnel consistait à maximiser ses récompenses à long terme. La valeur d'une politique traduira donc l'espérance de récompense à long terme de l'agent.

La politique optimale π^* est telle qu'elle maximise l'espérance de l'agent en tout état du système, soit :

$$\forall \pi \quad \forall s \in \mathcal{S} \quad V^{\pi^*}(s) \geq V^{\pi}(s)$$

Dans les problèmes à horizon fini $\mathcal{H}$, l'espérance de gain à partir d'un état s, et sur les $\mathcal{H}$ prochaines étapes, est considérée. Nous obtenons alors :

$$\forall s \in \mathcal{S} \quad V^{\pi}(s) = E\big[\sum_{t=0}^{\mathcal{H}} r_t | s_0 = s\big]$$

Si la fonction, de récompense est telle que $\mathcal{R} : \mathcal{S} \times \mathcal{A}$, alors $r_t = \mathcal{R}(s_t, \pi(s_t))$.

Dans le cas des problèmes à horizon infini, l'espérance de gain est pondérée par un facteur d'atténuation γ. D'où :

$$\forall s \in \mathcal{S} \quad V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$$

Cette formulation permet d'assurer la convergence de l'espérance. Cependant, la valeur du facteur d'atténuation doit être choisie avec précaution car elle influence le comportement de l'agent.

Une autre façon de déterminer la valeur d'une politique sur un horizon infini, consiste à calculer le gain moyen par étape de décision. Nous obtenons alors :

$$\forall s \in \mathcal{S} \quad V^{\pi}(s) = lim_{n \to \infty} E[\frac{1}{n} \sum_{t=0}^{n} r_t | s_0 = s]$$

Cette méthode de calcul s'affranchit du facteur d'atténuation mais elle a tendance à masquer les récompenses à court terme au profit des récompenses à long terme.

2.4.3 Le principe d'optimalité de Bellman

Bellman [Bellman, 1957] a montré que la fonction de valeur d'une politique peut être calculée par récurrence, grâce à l'équation de Bellman suivante :

$$V^{\pi}(s) = \mathcal{R}(s, \pi(s)) + \sum_{s' \in S} \gamma P(s, \pi(s), s') V_t^{\pi}(s')] \tag{2.3}$$

De plus, la fonction de valeur de la politique optimale d'un MDP satisfait l'équation de point fixe suivante :

$$V^*_{t+1}(s) = max_{a \in A}[\mathcal{R}(s,a) + \sum_{s' \in S} \gamma P(s,a,s') V^*_t(s')] \tag{2.4}$$

Cette équation se base sur le principe d'optimalité de Bellman selon lequel une trajectoire optimale à partir de l'étape $t+1$ est une sous-trajectoire optimale pour la politique optimale à l'étape t. Autrement dit, la politique optimale à l'étape $t+1$ est une sous-partie de la politique optimale à partir de l'étape t.

Un système d'équations peut alors être déduit en appliquant l'équation de Bellman à chaque état. Il a été montré que ce système admettait une solution unique à partir de laquelle une politique optimale peut être déduite. Bien que l'équation de Bellman admette une et une seule solution V^*, il peut exister plusieurs politiques optimales si celles-ci ont toutes pour valeur V^*.

L'équation de Bellman est à la base de plusieurs algorithmes de résolution des MDPs, en particulier Value Iteration et Policy Iteration que nous présenterons par la suite.

2.5 Une étude de complexité

Avant de décrire les deux principaux algorithmes utilisés pour la résolution des MDPs, nous allons nous intéresser à la complexité de la résolution optimale de ces problèmes. Nous commencerons par un bref rappel des différentes classes de complexité. Nous présenterons ensuite plusieurs résultats établissant la complexité des MDPs et POMDPs.

2.5.1 Les différentes classes de complexité

L'analyse de la complexité d'un problème permet d'étudier l'espace mémoire et le temps requis par un algorithme résolvant ce problème.

Les algorithmes P-complet peuvent être résolus en temps polynomial par une machine déterministe (machine de Turing par exemple). Soit n la taille en entrée du problème, une solution est obtenue dans le pire cas en $O(n^k)$ unités de temps (où k est une constante). Une opération élémentaire prenant une unité (pas) de temps, nous pourrons également exprimer la complexité d'un algorithmes en nombres d'opérations élémentaires.

Un algorithme NP-complet peut être résolu en temps polynomial par une machine non-déterministe. Une machine non-déterministe peut être vue comme une machine déterministe munie d'un « oracle » qui lui indiquerait quel chemin suivre pour arriver à la solution. Lors de

la recherche d'une solution, lorsque plusieurs choix sont possibles, une machine non-déterministe ferait automatiquement le bon. De telles machines n'existent pas. A ce jour, aucun algorithme ne peut simuler ce type de machine en temps polynomial. L'algorithme le plus performant pour une telle tâche est exponentiel en temps. Le passage d'une machine non-déterministe à une machine déterministe est donc exponentiel en temps. Un problème NP peut alors être résolu en temps exponentiel par une machine déterministe. En revanche, une solution à un problème NP peut être vérifiée en temps polynomial par une machine déterministe.

Étant donné qu'aucun algorithme polynomial connu ne peut résoudre un problème NP, la communauté scientifique suppose que les classes P et NP sont distinctes ($P \neq NP$). Cependant cette conjecture reste ouverte à de nombreuses recherches. Il est admissible d'affirmer qu'un problème NP n'est pas résoluble à part pour de très petites tailles de problème.

La complexité d'un problème peut également être évaluée en terme d'espace. La classe des problèmes PSPACE est constituée des problèmes P et NP pouvant être résolus en un espace polynomial. D'où $P \subset NP \subset PSPACE$.

D'autres classes de problèmes encore plus difficiles peuvent être répertoriées. Il s'agit des classes de complexité EXP (exponentielle déterministe) et NEXP (exponentielle non déterministe). Un problème de complexité EXP est un problème pouvant être résolu par une machine déterministe en temps exponentiel. Un problème NEXP est un problème nécessitant un temps exponentiel sur une machine non-déterministe, donc doublement exponentiel sur une machine déterministe.

Il a été montré que $EXP \subseteq NEXP$. On suppose cependant que $EXP \neq NEXP$, mais aucune preuve n'a été établie. En revanche, il est démontré que $P \neq EXP$.

L'étude de la complexité d'un problème peut apporter des informations précieuses sur la manière de le résoudre. En effet, s'il est possible d'affirmer qu'un problème est NP-complet (EXP ou NEXP), nous pouvons en déduire que la résolution exacte du problème n'est pas envisageable. Il est alors préférable de s'orienter vers la mise en œuvre d'algorithmes d'approximation.

2.5.2 La complexité des MDPs

Papadimitriou et Tsitsiklis [Papadimitriou et Tsitsiklis, 1987] ont étudié la complexité de la résolution optimale des MDPs et POMDPs.

Il a été démontré que la résolution d'un MDP à horizon fini ou infini est un problème P-complet. Il est donc possible de développer des algorithmes efficaces pour leur résolution.

La résolution d'un POMDP à horizon fini est un problème PSPACE, elle reste donc envisageable. En revanche, les POMDPs à horizon infini sont des problèmes indécidables. Littman s'est intéressé à un cas particulier de POMDP à horizon infini dans lequel les récompenses sont égales à 0 ou 1 (des valeurs booléennes). Il a montré que, dans ce cas, le problème est EXP.

2.6 Les algorithmes de résolution

Déterminer la politique optimale d'un MDP revient à calculer la valeur de la politique optimale. La politique optimale peut ensuite être déduite très facilement. Nous allons à présent décrire deux algorithmes pour la résolution de ces problèmes décisionnels.

Nous tenons à préciser que ces algorithmes supposent, tout comme les résultats présentés précédemment, que le modèle du système soit connu. En particulier, nous devons être en mesure de définir la fonction de transition $\mathcal{T}$ et la fonction de récompense $\mathcal{R}$.

Si l'agent évolue dans un environnement totalement inconnu, il n'est généralement pas possible de modéliser $\mathcal{T}$ et $\mathcal{R}$. L'hypothèse que nous avons implicitement posée n'est alors plus vérifiée. Dans ce cas, l'agent peut directement apprendre sa politique en utilisant des algorithmes d'apprentissage par renforcement tels que *Q-learning* qui ne nécessitent pas de connaissance *a priori* de $\mathcal{T}$ et $\mathcal{R}$. Pour plus de détails, le lecteur peut se référer à [Sutton et Barto, 1998, Kaelbling *et al.*, 1996] qui fournissent une introduction à ces mécanismes d'apprentissage.

2.6.1 L'algorithme Value Iteration

L'équation de Bellman 2.4 constitue la base de l'algorithme Value Iteration[18], l'un des algorithmes les plus utilisés pour la résolution de MDPs à horizons finis ou infinis.

Principe de l'algorithme

Cet algorithme de programmation dynamique décrit par Bellman [Bellman, 1957] consiste en une amélioration itérative de la valeur de chaque état du MDP. A partir d'une valeur initiale arbitraire de chaque état, Value Iteration améliore itérativement l'évaluation des états par approximations successives en utilisant l'équation 2.4. La valeur d'un état à l'itération t est calculée à partir de sa valeur à l'itération $t-1$. Le processus itératif s'arrête lorsque la différence entre les valeurs successives de tous les états est inférieure à un paramètre ϵ.

Étant donné que l'équation de Bellman est une équation de point fixe, nous sommes en mesure de garantir la convergence de l'algorithme vers une fonction de valeur optimale et ce quelle que soit la valeur initiale de la fonction de valeur. L'algorithme Value Iteration correspond alors à un algorithme de recherche de point fixe unique.

[18] Value Iteration est également appelé algorithme d'itération des valeurs.

Algorithme 1 : Value Iteration

Entrées : un MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$
un paramètre de précision ϵ
un facteur d'atténuation γ
une valeur initiale V_0 arbitraire

1 $t \leftarrow 0$
2 **pour tous les** $s \in \mathcal{S}$ **faire**
3 $V_0(s) \leftarrow V_o$
4 **fin**
5 **répéter**
6 $t \leftarrow t + 1$
7 **pour tous les** $s \in \mathcal{S}$ **faire**
8

$$V_t(s) = max_{a \in \mathcal{A}}[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s', a, s) V_{t-1}(s')]$$

9 **fin**
10 **jusqu'à** $max_{s \in \mathcal{S}} |V_t(s) - V_{t-1}(s)| < \epsilon$;
Sorties : V_n l'approximation de V^* à $\frac{2\epsilon\gamma}{1-\gamma}$ près

La valeur V_n de la politique obtenue par cet algorithme est une approximation à $\frac{2\epsilon\gamma}{1-\gamma}$ près de la valeur de la politique optimale. A partir de la valeur V_n, nous pouvons déduire une politique π_n optimale à ϵ près. En pratique, cette politique s'avère généralement être la politique optimale du MDP. Par ailleurs, nous pouvons constater que la politique optimale est souvent obtenue bien avant que l'algorithme ne s'arrête. En effet, les dernières itérations réalisent des améliorations de la fonction de valeur qui n'ont pour effet que d'augmenter la précision de l'évaluation mais ne modifient pas la politique.

Complexité

La complexité en temps de l'algorithme dépend de la complexité de chaque itération et du nombre d'itérations. Une itération nécessite $|\mathcal{A}||\mathcal{S}|^2$ opérations. Par ailleurs, le nombre d'itérations nécessaires pour converger est, au pire cas, polynomial en $|\mathcal{S}|$, $|\mathcal{A}|$, γ et B où B est le nombre total de bits requis pour représenter les données du problème.

Sutton et Barto [Sutton et Barto, 1998] ont montré que des MDPs d'un million d'états pouvaient facilement être résolus par l'algorithme Value Iteration. De plus, cet algorithme possède des propriétés anytime c'est-à-dire que la qualité de la solution augmente de façon monotone à chaque itération et l'amélioration est décroissante au cours du temps. L'algorithme est également

interruptible : si nous l'arrêtons avant qu'il ait atteint son point de convergence, nous disposerons quand même d'une solution.

Value Iteration calcule une politique ϵ-optimale pour n'importe quel état initial. Ceci peut constituer un avantage si nous souhaitons calculer une politique sans connaître *a priori* la situation initiale de l'agent. Cependant, lorsque les états initiaux sont connus, l'algorithme n'en tire pas avantage ce qui conduit à l'évaluation de trajectoires non atteignables.

En ce qui concerne la complexité en espace, Value Iteration est linéaire en $|\mathcal{S}||\mathcal{A}|$.

2.6.2 L'algorithme Policy Iteration

Policy Iteration[19] est également un algorithme itératif utilisant le principe d'optimalité de Bellman afin de résoudre des MDPs à horizon fini ou infini. Cet algorithme décrit par Howard en 1960 [Howard, 1960] procède par améliorations successives d'une politique.

Principe de l'algorithme

A la différence de Value Iteration qui calcule une valeur puis déduit une politique, Policy Iteration construit directement la politique optimale. L'algorithme part d'une politique initiale arbitraire π_0. A chaque itération, la politique courante est évaluée puis améliorée. L'évaluation utilise l'équation de Bellman 2.3 et revient à résoudre un système d'équations linéaires. La phase d'amélioration consiste à mettre à jour la politique courante en utilisant les valeurs calculées lors de la phase d'évaluation.

L'algorithme converge lorsqu'aucune modification de la politique n'est plus possible, soit $\pi_{t+1}(s) = \pi_t(s) \quad \forall s \in \mathcal{S}$. Par ailleurs, l'algorithme se termine en fournissant une politique optimale.

Howard a montré que chaque itération améliorait la politique : la valeur de chaque état à l'itération $t+1$ est supérieure ou égale à celle de l'itération t et elle est strictement supérieure pour au moins un état.

Complexité

Comme tout algorithme itératif, la complexité en temps de Policy Iteration dépend de la complexité d'une itération et du nombre d'itérations. L'évaluation d'une politique requiert $O(|S|^3)$ opérations élémentaires et la phase d'amélioration nécessite $O(|\mathcal{A}||\mathcal{S}|^2)$ opérations.

Le nombre d'itérations de l'algorithme dépend fortement de la politique initiale π_0. Il existe $|\mathcal{A}|^{|\mathcal{S}|}$ politiques possibles. Étant donné qu'à chaque itération la politique mise à jour domine strictement la politique précédente, le nombre d'itérations sera, au pire cas, exponentiel en nombre d'états. Toutefois, il a été prouvé que la borne supérieure concernant le nombre d'itérations était

[19] Policy Iteration est également appelé algorithme d'itération des politiques.

Algorithme 2 : Policy Iteration

Entrées : un MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$
un facteur d'atténuation γ
une politique initiale π_0 arbitraire

$t \leftarrow 0$
$\pi_{t+1} \leftarrow \pi_0$
répéter
  $\pi_t \leftarrow \pi_{t+1}$
  $t \leftarrow t + 1$
  `// phase d'évaluation`
  pour tous les $s \in \mathcal{S}$ **faire**
    Calculer $V_{\pi_t}(s)$
  fin
  `// phase d'amélioration`
  pour tous les $s \in \mathcal{S}$ **faire**
    si $\exists a \in \mathcal{A}$ *tel que* $\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') V_{\pi_t}(s') > V_{\pi_t}(s)$ **alors**
      $\pi_{t+1}(s) = a$
    sinon
      $\pi_{t+1}(s) = \pi_t(s)$
    fin
  fin
jusqu'à $\pi_t(s) = \pi_{t+1}(s)\ \forall s \in \mathcal{S}$;

Sorties : π_{t+1} la politique optimale

bien inférieure. En effet, Puterman [Puterman, 2005] a montré que l'algorithme ne converge pas moins vite que Value Iteration, pour les problèmes à horizon infini avec facteur d'atténuation. De plus, Value Iteration nécessite un nombre polynomial d'itérations pour résoudre ce type de problèmes. Il peut donc être déduit que le nombre d'itérations de Policy Iteration est polynomial pour des problèmes à horizon fini avec facteur d'atténuation. En revanche, l'influence du facteur d'atténuation sur le nombre d'itérations reste un problème ouvert.

En pratique, Policy Iteration converge en moins d'itérations que Value Iteration. En effet, l'algorithme s'arrête lorsque la politique optimale est atteinte. Value Iteration, au contraire, continue à itérer tant que la valeur de la politique peut être améliorée, et ce même si la politique ne change pas. En revanche, une itération de Policy Iteration consomme plus de temps qu'une itération de Value Iteration.

En ce qui concerne la complexité en espace, Policy Iteration est linéaire en $|\mathcal{S}||\mathcal{A}|$.

2.6.3 Discussion

Le choix d'utiliser Value Iteration ou Policy Iteration est difficile. Value Iteration procède à de petites améliorations successives alors que Policy Iteration réalise, à chaque itération, des améliorations plus conséquentes mais plus coûteuses en temps. Littman [Littman, 1996] avance que le choix de l'algorithme dépendrait de la structure du MDP. Néanmoins, aucun de ces deux algorithmes n'exploitent la structure du MDP afin d'améliorer ses performances [Littman *et al.*, 1995]. Ils restent ainsi très généraux et peuvent être utilisés pour tout type de MDP. Il est cependant suggéré qu'exploiter la structure du problème, permettrait de diminuer la complexité des algorithmes de résolution [Boutilier *et al.*, 1995]. Ainsi la connaissance de l'état initial peut par exemple être utilisée afin de restreindre le nombre de politiques à évaluer [Dean *et al.*, 1993]. De même, la structure des composantes du MDP peut être exploitée afin de réaliser une agrégation de l'espace d'états [Boutilier *et al.*, 1999, Boutilier et Dearden, 1994, Dean et Lin, 1995] ou d'actions [Hauskrecht *et al.*, 1998]. Dearden et Boutilier [Dearden et Boutilier, 1994] proposent également d'alterner la planification et l'exécution afin d'utiliser la connaissance de l'état courant pour décider de la prochaine action à exécuter. Toutes ces méthodes ont pour but de traiter plus efficacement des problèmes de grandes tailles.

Par ailleurs, Littman propose de diminuer les exigences quand à l'optimalité de la solution finale. Bien que les algorithmes que nous avons décrits soient polynomiaux, il n'est pas exclu de développer des algorithmes plus performants en temps ou en espace. Des méthodes d'approximation moins onéreuses peuvent ainsi être envisagées.

Comme nous le montrerons au chapitre suivant, des constatations similaires pourront être réalisées pour des modèles markoviens multi-agents. Elles constitueront le point de départ du travail que nous exposerons par la suite.

2.7 Les domaines d'application des MDPs

Dans l'introduction de son livre *Markov Decision Processes : discrete stochastic dynamic programming* [Puterman, 2005], Martin L. Puterman décrit la diversité des applications des MDPs. Il montre ainsi que les MDPs peuvent être utilisés dans des problèmes comme la gestion de stock, la maintenance du revêtement des routes, le remplacement d'équipements. Ce dernier exemple décrit par Howard dans le cadre du remplacement de pièces sur une voiture, peut également être adapté pour la maintenance d'un parc de bus de ville. Puterman propose également des applications en rapport avec les réseaux de communication : le problème étant alors d'établir un routage optimal des paquets à travers le réseau afin de maximiser la quantité d'information en transit. L'étude des phénomènes biologiques et écologiques offre également de grandes potentialités d'application des MDPs. Il est ainsi possible d'établir des modèles de comportement

de différents animaux (lions, oiseaux, araignées, ...). Enfin, les jeux de chance et de stratégie proposent un cadre naturellement adapté à l'utilisation des MDPs. Il est également possible de se référer à Mundhenk et al. [Mundhenk *et al.*, 2000] pour une description de la diversité des applications des MDPs.

Plus récemment les MDPs et POMDPs ont été utilisés en robotique [Kaelbling *et al.*, 1998]. A partir de l'utilisation des MDPs pour la gestion du dialogue [Young, 1990, Levin *et al.*, 1998], Roy et al. [Roy *et al.*, 2000] se sont intéressés à la mise en place, sur un robot, de dialogue parlé à l'aide de MDPs. Les MDPs et POMDPs ont également été utilisés afin de résoudre des problèmes de navigation [Simmons et Koenig, 1995, Gordon, 1999, Laroche, 2000]. En robotique exploratoire, nous pouvons citer les travaux de Bernstein et al. [Cardon *et al.*, 2001, Zilberstein *et al.*, 2002, Bernstein *et al.*, 2001] qui ont montré l'adéquation de ce formalisme pour la résolution des problèmes décisionnels auxquels est confronté un robot autonome évoluant sur Mars.

Conclusion

Dans ce chapitre, nous avons présenté différents modèles stochastiques permettant de raisonner sous incertitude, c'est-à-dire dans des environnements non-déterministes.

Nous avons tout d'abord présenté une propriété majeure de certains environnements stochastiques : la propriété de Markov. Nous avons montré que malgré les apparences, un grand nombre d'environnements pouvaient être modélisés de manière à vérifier cette propriété. Ceci nous a amenée à introduire les chaînes de Markov et leur adaptation à l'observabilité partielle : les chaînes de Markov cachées. Ces modèles permettent de représenter l'évolution de systèmes markoviens. Nous avons cependant mis en évidence le fait que ce formalisme ne permettait pas la modélisation de systèmes dans lesquels un agent influence, par l'exécution d'actions, l'évolution du système.

Les processus décisionnels de Markov ont alors été introduits. Nous avons passé en revue les différentes composantes (espace d'états, d'actions, fonction de transition et de récompense) d'un tel modèle. Nous nous sommes ensuite intéressée à la résolution de problèmes modélisés sous forme de MDPs. La notion de politique et le principe d'optimalité de Bellman ont été décrits, et nous avons détaillé deux algorithmes majeurs pour la résolution des MDPs. L'optimalité et la complexité de ces algorithmes ont enfin été discutés.

Tout au long de ce chapitre, nous ne nous sommes pas préoccupée de l'aspect multi-agent des systèmes considérés. Les processus décisionnels de Markov ont en effet été introduits comme des formalismes permettant de représenter des problèmes décisionnels mono-agent, alors que nous avions souligné, au chapitre précédent, l'importance de l'aspect multi-agent dans notre travail. Nous pouvons donc nous interroger sur la façon de formaliser des problèmes décisionnels multi-agents à l'aide de MDPs. Nous répondrons à cette question dans le chapitre suivant.

Chapitre 3

Les Processus Décisionnels de Markov et les SMA

Dans les chapitres qui ont précédé, nous avons présenté d'un côté les Systèmes Multi-Agents (SMA), et de l'autre les Processus Décisionnels de Markov (MDP). Ce chapitre va porter sur la rencontre de ces deux champs d'étude. Nous allons ainsi traiter de la résolution des problèmes de décision séquentielle dans les systèmes multi-agents. Nous aborderons plus particulièrement l'utilisation des processus décisionnels de Markov pour la résolution des problèmes de décision multi-agents.

Nous commencerons par présenter la problématique de la prise de décision sous incertitude dans les systèmes multi-agents. Nous en préciserons les éléments essentiels et nous mettrons en relation la théorie des jeux et les problèmes décisionnels multi-agents.

Nous présenterons ensuite le formalisme des processus décisionnels de Markov multi-agents et nous discuterons de leur adéquation pour résoudre les problèmes qui nous sont posés. Cette discussion nous amènera à définir le formalisme des processus décisionnels de Markov décentralisés (DEC-POMDP). Nous en énumérerons les propriétés majeures et nous identifierons un certain nombre de classes de DEC-POMDPs.

Enfin, nous procéderons à un état de l'art des différentes méthodes existantes pour la résolution des processus décisionnels de Markov décentralisés. Après avoir présenté les méthodes de résolution exactes et approchées, nous discuterons de leur efficacité et de leur applicabilité.

3.1 Décision séquentielle sous incertitude dans les SMA

Nous nous intéressons à la prise de décision séquentielle sous incertitude dans les systèmes multi-agents. Nous considérons un ensemble de n agents, chaque agent $\mathcal{A}g_i$ pouvant réaliser un ensemble A_i d'actions a_i. A chaque instant t, chaque agent doit décider quelle action réaliser afin

de maximiser ses performances.

Un problème de décision séquentielle peut être découpé en étapes de décision. Les problèmes formalisés au chapitre précédent par des MDPs constituaient des problèmes de décision séquentielle. Dans le contexte multi-agent, à chaque étape, les agents doivent décider quelles actions exécuter.

3.1.1 Caractéristiques du problème

Considérons deux agents devant déplacer un cube d'un point A à un point B. Nous représenterons le monde dans lequel évoluent les agents par une grille 5×5. Le cube ne peut être déplacé que si les agents le poussent en même temps. Les déplacements du cube sont illustrés par la figure 3.1. Si l'un des agents pousse le cube et que l'autre ne fait rien ou se déplace, le cube ne bouge pas. Si les deux agents poussent le cube, le déplacement est incertain, le cube peut rester sur place ou dévier. Les probabilités indiquées sur la figure 3.1 ne rendent compte que des cas où le cube bouge. Lorsque la somme des probabilités est inférieure à 1, la différence fournit la probabilité que le cube reste sur place.

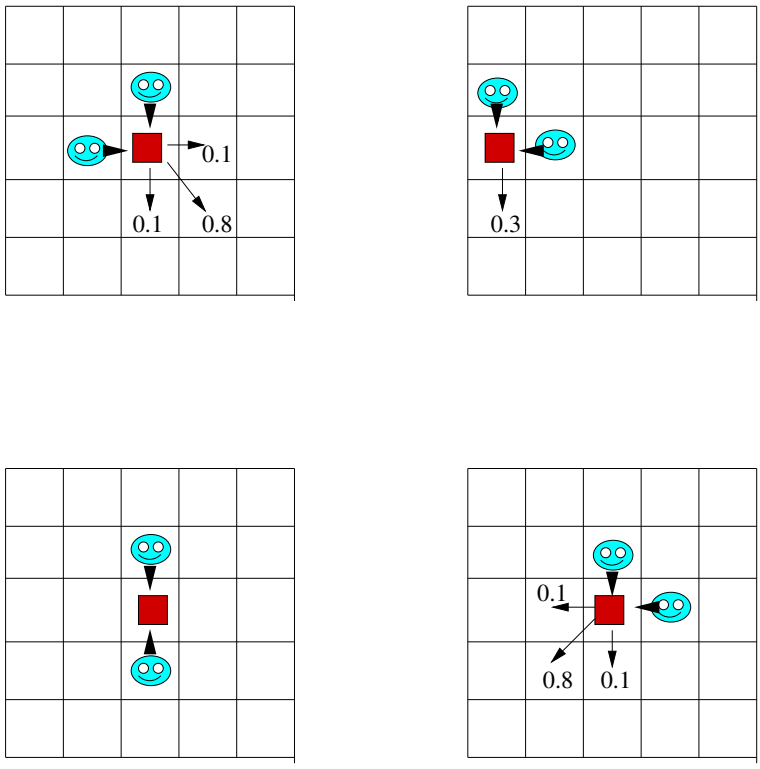


FIG. 3.1 – Déplacement d'un cube poussé par deux agents

Comme nous l'avons décrit précédemment, les agents n'ont généralement qu'une vue partielle de leur état local et de l'état du système. De plus, chaque agent ne connaît pas le comportement

des autres agents et doit donc prendre une décision à partir d'une connaissance partielle du système. Dans notre exemple, les perceptions de chaque agent peuvent être restreintes à sa position. De ce fait, chaque agent ne connaît ni la position, ni la décision de l'autre agent. Il doit donc prendre ses décisions en dépit de ce manque de connaissance.

Nous avons montré au premier chapitre qu'il était impossible de rendre compte de tous les paramètres régissant un système réel. L'incertitude est donc utilisée afin de pallier cette incomplétude et est modélisée par une fonction de transition décrivant une distribution de probabilités sur les résultats des actions.

Dans les systèmes coopératifs, le but des agents est de maximiser leur récompense jointe. Pour ce faire, il est nécessaire que les agents se coordonnent. Étant donné que les agents partagent une fonction de récompense jointe, la décision d'un agent a une influence sur tous les autres membres du système. Un manque de coordination risque de mener les agents à un comportement ne permettant pas de maximiser leurs performances.

Dans notre exemple, il est nécessaire que les agents poussent en même temps dans la même direction sinon, le cube ne bougera pas et aucune récompense ne sera obtenue.

Notons que dans le cas des systèmes non-coopératifs, chaque agent doit également anticiper les actions des autres agents afin de maximiser son gain propre.

3.1.2 Notion de politique

Résoudre un problème de décision séquentielle multi-agent consiste à calculer une politique jointe $\pi = \langle \pi_1, \cdots, \pi_n \rangle$ où π_i correspond à la politique de l'agent $\mathcal{A}g_i$. Cette dernière permet à l'agent de décider comment agir : elle fait correspondre une action à chaque situation dans laquelle l'agent peut se trouver. Les politiques sont également appelées stratégies. Un vecteur de stratégies est alors équivalent à une politique jointe et désigne le tuple $\pi = \langle \pi_1, \cdots, \pi_n \rangle$.

Deux types de politiques peuvent par ailleurs être identifiées [Xuan et Lesser, 2002] : les politiques centralisées et les politiques décentralisées. La différence réside dans la manière dont elles seront exécutées.

Politiques centralisées et décentralisées

Une politique centralisée spécifie, pour chaque agent, l'action qu'il devra exécuter en fonction de l'état du système. Il s'agit d'une fonction $\pi = \langle \pi_1, \cdots, \pi_n \rangle$ telle que $\pi_i : \mathcal{S} \rightarrow \mathcal{A}_i$.

Une politique décentralisée spécifie, pour chaque agent, quelle action réaliser en fonction des connaissances que l'agent a de l'état global du système. De ce fait, les agents peuvent ne pas avoir accès à l'état du système. A partir des connaissances de l'agent (les observations et les états de croyances), la politique dicte quelle action réaliser. Ce type de politique est beaucoup mieux adapté dans le cas d'agents répartis dans l'espace et autonomes, c'est-à-dire prenant leurs

décisions sans intervention d'un contrôleur central. Une politique décentralisée fait correspondre à chaque état s_i de l'agent $\mathcal{A}g_i$, une action a_i. Lorsque l'agent n'observe pas totalement son état s_i, la politique est définie sur l'ensemble des observations Ω_i de l'agent. Elle correspond à une fonction $\pi_i : \Omega_i \rightarrow \mathcal{A}_i$.

La politique jointe des agents est généralement calculée avant que les agents commencent l'exécution de leurs tâches. La phase précédant l'exécution et durant laquelle le calcul des politiques est réalisé, est appelée phase « hors-ligne ». La phase d'exécution des tâches correspond à la phase « en ligne ». Déterminer hors-ligne les politiques des agents permet de limiter les calculs que les agents auront à réaliser durant l'exécution des tâches. De ce fait, les agents possèdent une plus grande réactivité.

Politique optimale

Une politique optimale jointe π^* est telle qu'elle maximise l'espérance de gain des agents. Nous désignerons par politique π_i^*, la politique de l'agent $\mathcal{A}g_i$ composant la politique optimale jointe $\pi^* = \langle \pi_1^*, \cdots, \pi_n^* \rangle$.

Dans le cadre d'agents coopératifs, la fonction de récompense est commune à tous les agents. Maximiser le gain espéré d'un agent revient à maximiser le gain espéré de tous les agents.

Plus formellement, dans un cadre coopératif, les agents tentent de maximiser leur espérance définie par l'équation suivante :

$$E[\sum_{t=0}^{\mathcal{H}-1} \mathcal{R}(s_t, a_t, s_{t+1})|s_0]$$

où $\mathcal{H}$ désigne l'horizon du problème, s_t est l'état du système à l'instant t et a_t désigne l'action jointe exécutée par les agents à l'instant t. s_0 correspond à l'état initial du système. $\mathcal{R}(s_t, a_t, s_{t+1})$ est la récompense obtenue par les agents lorsque le système passe d'un état s_t à un état s_{t+1} en effectuant l'action a_t.

La valeur d'une politique étant mesurée à partir du critère ci-dessus, nous obtenons :

$$V^{\pi}(s_0) = E[\sum_{t=0}^{\mathcal{H}-1} \mathcal{R}(s_t, \pi(s_t), s_{t+1})|s_0]$$

En supposant l'état initial s_0 connu, la politique optimale π^* est telle que :

$$V^{\pi}(s_0) = arg\_max_{\pi} V^{\pi}(s_0)$$

Dans les systèmes non-coopératifs, chaque agent possède sa propre fonction de récompense, nous obtenons alors la mesure suivante :

$$V^{\pi}(s_0) = E[\sum_{t=0}^{\mathcal{H}-1} \mathcal{R}_i(s_t, \pi(s_t), s_{t+1})|s_0] \tag{3.1}$$

Dans le cadre de problèmes à horizons infinis, l'espérance peut être pondérée par un facteur d'atténuation. Une moyenne des gains espérés peut également être envisagée.

3.2 Théorie des jeux et décision sous incertitude

Le premier ensemble cohérent de résultats concernant la **théorie des jeux** a été formulé par Von Neuman et Morgenstern dans un ouvrage intitulé *The theory of Games and Economic Behavior* [Von Neumann et Morgenstern, 1944]. La théorie des jeux s'inspire de concepts provenant des mathématiques et des sciences économiques, et propose de **modéliser par un formalisme abstrait les interactions entre des décideurs** (agents) rationnels. Un jeu est composé d'un ensemble de joueurs, de règles et d'une fonction de récompense. Chaque joueur doit alors établir une stratégie (ou politique) lui permettant de décider comment agir à chaque tour de jeu afin de maximiser sa récompense. Il existe différents types de jeux : les joueurs peuvent être coopératifs ou non-coopératifs, les joueurs peuvent jouer simultanément ou de manière séquentielle, l'information dont dispose chaque joueur peut être complète ou incomplète, parfaite ou imparfaite.

3.2.1 Jeux de Markov

Dans un jeu à n-joueurs classique, l'action entreprise par un joueur à l'étape t (ou tour de jeu t) n'a d'influence que sur le paiement qu'il obtient à cette étape (environnement épisodique). La décision prise par le joueur n'a donc aucune répercussion sur les paiements futurs ou sur les possibilités de jeu à venir. Dans de nombreuses situations d'interactions entre des joueurs, cette hypothèses s'avère fausse. Prenons l'exemple de deux parieurs possédant une fortune de départ. A chaque tour, chaque parieur met en jeu une certaine somme (inférieure ou égale à sa fortune). Le joueur ayant parié la plus grosse somme remporte toutes les sommes mises en jeu. La somme pariée à chaque tour a une influence sur les possibilités de pari et de gain futur des joueurs.

Shapley [Shapley, 1953] a défini les jeux stochastiques afin de tenir compte de ce type de phénomène. Un espace d'états $\mathcal{S}$ et une probabilité de transition $\mathcal{P}$ ont ainsi été ajoutés à la définition des jeux classiques.

Définition 10 ***Jeu Stochastique***

Un jeu stochastique est défini par un tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{T}, u_1, \cdots, u_n \rangle$ *tel que :*

- $\mathcal{S}$ *est l'ensemble fini des états du jeu*
- $\mathcal{N}$ *définit l'ensemble des n joueurs*
- $\mathcal{A} = \langle \mathcal{A}_1, \cdots, \mathcal{A}_n \rangle$ *est l'ensemble des actions des joueurs.* $\mathcal{A}_i$ *est l'ensemble des actions* a_i *exécutables par le joueur* $\mathcal{A}g_i$.

- $\mathcal{P}$ *définit la fonction de transition du jeu.* $\mathcal{P}(s, a, s')$ *donne la probabilité que le système passe d'un état* s *à un état* s' *lorsque les joueurs exécutent l'action* $a =< a_1, \cdots, a_n >$
- $u_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ *correspond à la fonction d'utilité du joueur* $\mathcal{A}g_i$.

Dans un jeu stochastique, le choix de l'action réalisée par un joueur à l'étape t influence son utilité $u_i(s^t, a^t)$ à cette étape. Cette décision a également des répercussions sur les états suivants s^{t+1} possibles et sur les paiements futurs des joueurs.

La fonction de transition d'un jeu stochastique vérifie la propriété de Markov, c'est pourquoi les jeux stochastiques sont également appelés jeux de Markov.

3.2.2 Notion d'équilibre

Dans un jeu stochastique, chaque joueur cherche à maximiser son espérance de gain. La valeur d'un vecteur de stratégie π, pour un agent $\mathcal{A}g_i$, est définie comme étant l'espérance de gain de l'agent, soit :

$$V_i^{\pi} = E[\sum_{t=0}^{\infty} \gamma^t u_i(s^t, \pi(s^t))|s_0]$$

où γ correspond au facteur d'atténuation. Calculer V_i^{π} revient à calculer l'espérance des trajectoires possibles $s_0, s_1, ...$ Dans le cas d'un horizon fini, la somme est réalisée sur un nombre fini d'étapes et le facteur d'atténuation est omis, on se ramène alors à l'équation 3.1.

Parmi l'ensemble des vecteurs de stratégies possibles, on peut identifier des situations d'équilibres comme l'équilibre de Nash[20] [Nash, 1950].

Définition 11 ***Équilibre de Nash***

Un vecteur de stratégies $\pi = \langle \pi_1, \cdots, \pi_n \rangle$ *est un équilibre de Nash si :*

$$\forall i,\ V_i^{\pi}(s) \geq V_i^{\pi_{-i} \times \pi'_i}(s)\ \forall s \in \mathcal{S}\ \forall \pi'_i$$

où π_{-i} *désigne la politique des agents* $\mathcal{A}g_j$ *tels que* $j \neq i$.

Un vecteur de stratégies est un équilibre de Nash si aucun joueur n'a intérêt à dévier unilatéralement de l'équilibre. Étant fixées les stratégies de tous les joueurs $j \neq i$, $\mathcal{A}g_i$ ne peut que diminuer son gain espéré s'il modifie sa stratégie. Un équilibre de Nash représente un optimum local.

Les vecteurs de stratégies correspondant à des équilibres de Nash constituent des solutions intéressantes aux problèmes de décision à n-joueurs. Hors d'un équilibre de Nash, au moins un des joueurs tirera profit de la modification de sa politique. Les théoriciens s'accordent sur le fait qu'une solution est nécessairement un équilibre de Nash. Cependant, dans de nombreux jeux, il existe plusieurs équilibres de Nash. Se pose alors le problème du choix de l'équilibre.

[20] Notons que les équilibres de Nash sont également appelés équilibres parfaits de Markov.

3.2.3 Résolution de jeux de Markov

Les travaux menés sur la résolution des jeux Markoviens sont essentiellement axés sur les jeux non-coopératifs.

Littman [Littman, 1994] a exploré la possibilité de résoudre des jeux de Markov à deux joueurs et à somme nulle, en utilisant des algorithmes d'apprentissage par renforcement. Le fait de considérer des jeux à deux joueurs et à somme nulle, simplifie le modèle mathématique. Cependant, cette hypothèse entraîne d'importantes restrictions sur les possibilités de modélisation des systèmes multi-agents. De plus, il devient impossible de prendre en compte des phénomènes de coopération entre les joueurs.

Hu et Wellman [Hu et Wellman, 1998a, Hu et Wellman, 1998b] se sont intéressés aux jeux de Markov à somme non-nulle et à n agents. Bien que ce type de jeu permette à des agents d'adopter des comportement coopératifs, Hu et Wellman n'ont traité que le cas d'agents non-coopératifs.

Dans le cadre de cette thèse, nous nous intéresserons à des groupes d'agents coopératifs. Dans la suite de ce chapitre, nous allons donc nous concentrer sur les approches permettant de résoudre des problèmes décisionnels pour de tels systèmes.

3.3 Processus Décisionnels de Markov Multi-agents

Afin d'adapter le formalisme des MDPs aux systèmes multi-agent coopératifs, Boutilier [Boutilier, 1996, Boutilier, 1999, Boutilier *et al.*, 1999] a défini les Processus Décisionnels de Markov Multi-agent ou MMDP (Multiagent Markov Decision Processes). Ces derniers permettent de formaliser des problèmes de décision séquentielle dans des systèmes multi-agents coopératifs. Ce formalisme est très proche de celui des jeux de Markov. Cependant, les MMDP modélisent uniquement des systèmes coopératifs. La fonction de récompense est donc définie communément à tous les joueurs, alors que les jeux stochastiques définissent une fonction de récompense propre à chaque agent.

3.3.1 Formalisme

Un MMDP est défini par un tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ tout comme les processus décisionnels de Markov classiques. Cependant, chaque action est décrite par l'ensemble des actions individuelles des agents, on parle alors d'action jointe. En plus du tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$, une variable α est définie. Elle correspond au nombre d'agents du système.

Définition 12 ***Processus Décisionnels de Markov Multi-agents***

Un MMDP est défini par un tuple $< \alpha, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ *tel que :*

- α *est le nombre d'agents du système.*
- $\mathcal{S}$ *correspond à l'ensemble des états* s *du système.*
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ *définit l'ensemble des actions jointes des agents,* $\mathcal{A}_i$ *est l'ensemble des actions locales de l'agent* $\mathcal{A}g_i$*.*
- $\mathcal{P}$ *est une fonction de transition, elle donne la probabilité* $\mathcal{P}(s, a, s')$ *que le système passe dans un état* s' *quand les agents exécutent l'action jointe* $a \in \mathcal{A}$ *à partir de l'état* s*.*
- $\mathcal{R}$ *définit la fonction de récompense.* $\mathcal{R}(s, a, s')$ *est la récompense obtenue par le système lorsqu'il passe d'un état* s *à un état* s' *en exécutant l'action* a*.*

Un MMDP peut être vu comme un MDP ayant un grand espace d'états et d'actions. L'ensemble des agents est alors considéré comme un seul agent dont le but est de calculer une politique optimale pour le MDP joint. Un MMDP peut également être considéré comme un jeu stochastique à n-joueurs dans lequel la fonction de récompense est la même pour tous les joueurs. Le formalisme des MMDPs correspond donc à une généralisation des MDP au cas multi-agent et à une spécialisation des jeux stochastiques à n-joueurs.

Résoudre un MMDP consiste à calculer une politique jointe $\pi = \langle \pi_1, \cdots, \pi_n \rangle$ où π_i correspond à la politique locale de l'agent $\mathcal{A}g_i$. Elle définit une fonction $\pi_i : \mathcal{S} \rightarrow \mathcal{A}_i$ qui fait correspondre à tout état du système, une action a_i de l'agent $\mathcal{A}g_i$. Une telle politique jointe peut être calculée par un algorithme classique comme Value Iteration.

La politique d'un MMDP est une politique centralisée, il est nécessaire pour l'exécuter que chaque agent ait accès à l'état global du système. Dans les systèmes multi-agents cette hypothèse est rarement vérifiée. Afin d'exécuter la politique du MMDP, il est alors nécessaire que le contrôle soit centralisé ou bien que les agents puissent communiquer.

3.3.2 Contrôle Centralisé

Les systèmes multi-agents à contrôle centralisé sont conçus de telle façon qu'un contrôleur central est chargé de calculer la politique optimale jointe et de prendre les décisions pour tous les agents. Dans de tels systèmes, la politique d'un MMDP peut être facilement exécutée. L'entité centrale connaît l'état du système et peut dicter à chaque agent quelle action réaliser.

Un tel type de contrôle n'est généralement pas envisageable. En effet, les agents sont souvent répartis dans l'espace et aucun n'a assez d'informations pour jouer le rôle d'entité centrale.

3.3.3 Contrôle Décentralisé

Appliquer la politique d'un MMDP de manière décentralisée est cependant possible si les agents peuvent communiquer à volonté et gratuitement, c'est-à-dire si la communication n'est

pas limitée par des contraintes physiques telles que la taille de la bande passante, et qu'elle n'a aucune influence sur l'utilité des agents. Ces derniers peuvent alors échanger leurs informations sur leurs états afin de déterminer l'état global du système. Il est malgré tout nécessaire que l'état du système soit collectivement totalement observable. Sinon, les agents ne pourront pas déduire l'état du système même s'ils ont échangé toutes leurs informations.

Dans les systèmes multi-agents où le contrôle est décentralisé, le calcul de la politique peut être réalisé de manière centralisée ou décentralisée. Dans le premier cas, un contrôleur central est chargé de calculer la politique optimale jointe $\pi = \langle \pi_1, \cdots, \pi_n \rangle$. Les politiques locales π_i constituant cette politique optimale jointe sont ensuite envoyées aux agents chargés de les exécuter.

Un autre modèle de conception, complètement décentralisé, peut être utilisé. Chaque agent calcule alors de manière individuelle la politique optimale jointe. Si les agents suivent le même raisonnement pour effectuer ce calcul, ils obtiendront tous la même politique optimale. Cependant, s'il existe plusieurs politiques optimales, il est nécessaire que les agents coordonnent le choix de la politique optimale à suivre. Dans le cas contraire, des problèmes de coordination risquent de survenir. En effet, s'il existe plusieurs politiques optimales, les agents peuvent sélectionner différentes politiques et ainsi faire des choix non-coordonnés qui conduiront à un comportement non-optimal.

Considérons deux agents devant se déplacer sur une route à deux voies. Ce problème est représenté par la figure 3.2. Chaque agent peut soit rouler à droite, soit rouler à gauche. Les agents roulent en sens inverse et doivent se croiser en évitant toute collision. Pour ce faire, ils doivent rouler sur des voies différentes. Si les agents roulent du même côté (donc sur des voies différentes), ils obtiennent une récompense de +1, sinon il obtiennent une pénalité de -1. Il existe quatre politiques jointes possibles :

- L'agent 1 roule à droite et l'agent 2 roule à gauche.
- L'agent 1 roule à gauche et l'agent 2 roule à droite.
- Les deux agents roulent à gauche.
- Les deux agents roulent à droite.

Deux de ces politiques sont des politiques optimales jointes : tous les agents roulent à droite ou tous les agents roulent à gauche. Cependant, si le premier agent choisit la politique optimale jointe « les deux agents roulent à gauche », et le second agent sélectionne « les deux agents roulent à droite », il en résultera un comportement non-optimal.

Afin que les agents aient un comportement optimal, il est donc nécessaire qu'ils sélectionnent de manière coordonnée la politique optimale jointe à suivre. Pour ce faire, différentes alternatives

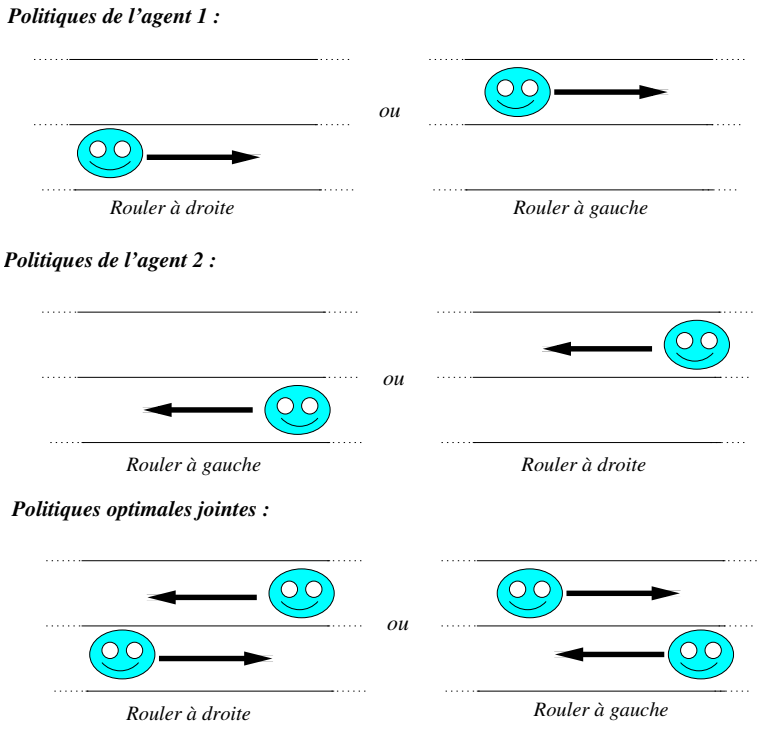


FIG. 3.2 – Problème de croisement à deux agents

peuvent être envisagées : la communication, l'application de conventions ou de lois sociales, ou bien l'apprentissage.

En communiquant les agents peuvent s'échanger des messages et aboutir à un accord sur la politique à sélectionner.

Les lois sociales et les conventions décrivent un critère de choix pour les agents. Dans le problème précédent, afin de sélectionner la politique optimale jointe, les agents peuvent utiliser des conventions telles que « En France, on roule à droite. ». Ainsi, les agents sélectionneront tous la première politique optimale jointe.

L'apprentissage permet aux agents d'apprendre par essais-erreurs quelle politique jointe sélectionner. Si lors de l'application d'une politique, un agent n'obtient pas la récompense escomptée, il en déduit que les autres agents n'ont pas sélectionné cette politique jointe. Il pourra alors décider de sélectionner une autre politique optimale au prochain essai.

En décrivant les MMDP, Boutilier a proposé un premier formalisme intégrant systèmes multi-agents coopératifs et Processus Décisionnels de Markov. L'utilisation des Processus Décisionnels de Markov Multi-agents (MMDP) suppose l'existence d'un contrôleur central ayant une vue globale du système et prenant des décisions de façon centralisée. Cependant, dans la plupart des

systèmes multi-agents, chaque agent n'a qu'une vue locale de l'état global du système sur laquelle il base ses décisions. Les MMDP posent donc des hypothèses contraignantes qui empêchent leur utilisation dans bon nombre de systèmes multi-agents.

3.4 Contrôle décentralisé et Processus Décisionnels de Markov

3.4.1 Formalisme des DEC-POMDPs

Les Processus Décisionnels de Markov Décentralisés Partiellement Observables (DEC-POMDP) et les Processus Décisionnels de Markov Décentralisés (DEC-MDPs) constituent des extensions des POMDPs et des MDPs pour des domaines où le contrôle est décentralisé.

Les Processus Décisionnels de Markov décentralisés

Ces formalismes permettent la modélisation de problèmes de décision multi-agents dans lesquels des agents doivent réaliser un ensemble de tâches en maximisant leur récompense globale. De plus, chaque agent doit décider de manière autonome quelle action exécuter tout en n'ayant qu'une vue partielle du système.

Les DEC-MDPs et les DEC-POMDPs définissent un ensemble d'observations Ω constitué des observations locales Ω_i des agents. Une fonction d'observation $\mathcal{O}$ est également définie, elle décrit la probabilité qu'un agent $\mathcal{A}g_i$ observe o_i étant donnés un état de départ s, une action jointe a et un état d'arrivée s'.

Définition 13 ***Les Processus Décisionnels de Markov Décentralisés Partiellement Observables***

Un DEC-POMDP est défini par un tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \Omega, \mathcal{O}, \mathcal{R} >$ *tel que :*

- $\mathcal{S}$ *définit l'ensemble des états du système.*
- $\mathcal{A} = \langle \mathcal{A}_1, \cdots, \mathcal{A}_n \rangle$ *est l'ensemble des actions jointes.* $\mathcal{A}_i$ *définit l'ensemble des actions* a_i *de l'agent* $\mathcal{A}g_i$*.*
- $\mathcal{P} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ *définit la fonction de transition.* $\mathcal{P}(s, a, s')$ *correspond à la probabilité que le système passe d'un état* s *à un état* s' *lorsque l'action jointe* a *est exécutée.*
- $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$ *est l'ensemble des observations des agents et* Ω_i *est l'ensemble des observations* o_i *de l'agent* $\mathcal{A}g_i$*.*
- $\mathcal{O} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0,1]$ *définit la fonction d'observation.* $\mathcal{O}(s, a, s', o = \langle o_1, \cdots, o_n \rangle)$ *correspond à la probabilité que chaque agent* $\mathcal{A}g_i$ *observe* o_i *lorsque les agents exécutent l'action jointe* a *à partir de l'état* s *et que le système arrive dans l'état* s'*.*
- $\mathcal{R}$ *définit la fonction de récompense.* $\mathcal{R}(s, \langle a_1, \cdots, a_n \rangle, s')$ *est la récompense obtenue par le système lorsque les agents exécutent l'action* $\langle a_1, \cdots, a_n \rangle$ *à partir de l'état* s *et arrivent dans l'état* s'*.*

La fonction de transition ainsi que la fonction de récompense dépendent de l'action jointe exécutée par tous les agents. Comme les POMDPs, les DEC-POMDPs sont définis sur un horizon $\mathcal{H}$, fini ou infini. A chaque exécution d'une action jointe, la fonction $\mathcal{R}$ détermine la récompense octroyée au système ; le but des agents étant de maximiser la somme des récompenses obtenues. Notons qu'un DEC-POMDP à un seul agent est équivalent à un POMDP.

Si l'état global du système peut être déduit à partir des observations locales cumulées des agents, le DEC-POMDP devient un DEC-MDP. L'état global du système est alors collectivement totalement observable. Cette propriété peut se traduire plus formellement de la façon suivante :

$$\text{si } \mathcal{O}(s, a, s', o = \langle o_1, \cdots, o_n \rangle) > 0 \text{ alors } Prob(s' | \langle o_1, \cdots, o_n \rangle) = 1$$

Remarquons que cette propriété n'implique pas nécessairement que chaque agent observe totalement son état local. De plus, l'état global du système n'est pas pour autant individuellement observable (un agent n'observe pas pour autant l'état global du système). Un DEC-MDP présente donc les mêmes composantes qu'un DEC-POMDP.

Dans le cas où les états des agents sont localement totalement observables, c'est-à-dire que $\forall o_i \in \Omega_i \quad \exists s_i \in \mathcal{S}_i \quad P(s_i = o_i) = 1$, l'ensemble des observations Ω et la fonction d'observation $\mathcal{O}$ peuvent être omis afin d'éviter les redondances d'information. Le DEC-MDP est alors défini par l'ensemble $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$.

Multi-agent Team Decision Problem

Pynadath et Tambe [Pynadath et Tambe, 2002] ont décrit un formalisme plus général que les DEC-POMDPs : le Multiagent Team Decision Problem (MTDP). Tout comme les DEC-POMDPs, les MTDPs permettent de formaliser des problèmes de contrôle décentralisé sous incertitude. Un MTDP reprend les mêmes composantes qu'un DEC-POMDP. Afin de représenter plus facilement les problèmes de contrôle dans les systèmes multi-agents, un ensemble d'états de croyances est ajouté à la définition des DEC-POMDPs. L'état de croyances b_i^t d'un agent $\mathcal{A}g_i$ décrit son état mental à l'instant t. Il est ainsi possible de différencier croyances et observations. En effet, dans le formalisme des DEC-POMDPs, l'état de croyances d'un agent $\mathcal{A}g_i$ correspond à ses observations et la politique locale est une fonction $\pi_i : \Omega_i \rightarrow \mathcal{A}_i$ qui fait correspondre une action à chaque observation de l'agent $\mathcal{A}g_i$. L'introduction des états de croyances dans la formalisation du problème, permet une modélisation plus riche de l'état mental de l'agent qui n'est plus seulement limité aux observations. La politique d'un agent devient alors une fonction $\pi_i : B_i \rightarrow \mathcal{A}_i$ qui fait correspondre une action à chaque état de croyances de l'agent $\mathcal{A}g_i$.

Définition 14 ***Multi-agent Team Decision Problem***

Un MTDP est défini par un tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, B_1, \cdots, B_n, \Omega, \mathcal{O}, \mathcal{R} >$ *tel que :*

- $\mathcal{S}$ *définit l'ensemble des états du système.*
- $\mathcal{A} = \langle \mathcal{A}_1, \cdots, \mathcal{A}_n \rangle$ *est l'ensemble des actions jointes et* $\mathcal{A}_i$ *définit l'ensemble des actions* a_i *de l'agent* $\mathcal{A}g_i$*.*
- $\mathcal{P} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ *définit la fonction de transition.* $\mathcal{P}(s, a, s')$ *correspond à la probabilité que le système passe d'un état* s *à un état* s' *lorsque l'action jointe* a *est exécutée.*
- B_i *désigne l'ensemble des états de croyances de l'agent* $\mathcal{A}g_i$*. Un tel ensemble est défini pour chaque agent. Un agent* $\mathcal{A}g_i$ *construit son état de croyances* b_i^t *à l'instant* t *à partir de ses observations jusqu'à l'instant* t*.*
- $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$ *est l'ensemble des observations des agents tel que* Ω_i *est l'ensemble des observations de l'agent* $\mathcal{A}g_i$*.*
- $\mathcal{O} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0, 1]$ *définit la fonction d'observation.* $\mathcal{O}(s, a, s', o = \langle o_1, \cdots, o_n \rangle)$ *correspond à la probabilité que chaque agent* $\mathcal{A}g_i$ *observe* o_i *lorsque les agents exécutent l'action jointe* a *à partir de l'état* s *et que le système arrive dans l'état* s'*.*
- $\mathcal{R}$ *définit la fonction de récompense.* $\mathcal{R}(s, \langle a_1, \cdots, a_n \rangle, s')$ *est la récompense obtenue par le système lorsque les agents exécutent l'action* $\langle a_1, \cdots, a_n \rangle$ *à partir de l'état* s *et arrivent dans l'état* s'*.*

Lorsque l'état de croyances d'un agent se limite à ses observations, le MTDP est réduit à un DEC-POMDP.

Les relations entre les MTDPs, DEC-POMDPs, DEC-MDPs, POMDPs et MDPs peuvent être résumées par le diagramme de la figure 3.3 et le tableau 3.1.

Type de contrôle	Centralisé		Décentralisé	
État global collectivement totalement observable	Oui	Non	Oui	Non
Formalisme	MDP	POMDP	DEC-MDP	DEC-POMDP

TAB. 3.1 – Relations entre les DEC-POMDPs, DEC-MDPs, POMDPs et MDPs

Dans le cadre du contrôle centralisé, observabilité locale et observabilité collective sont équivalentes. Un MDP est donc localement totalement observable. En revanche un DEC-MDP n'est pas nécessairement localement totalement observable.

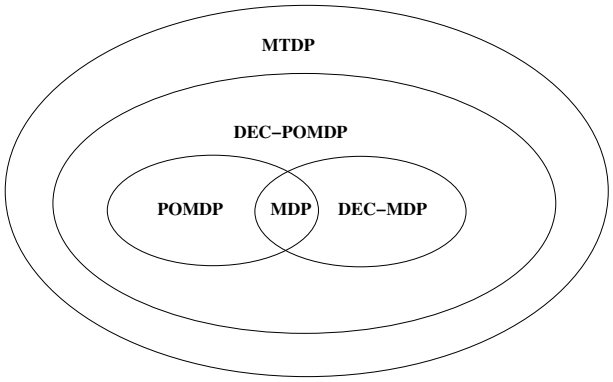


FIG. 3.3 – Relations entre les différents types de Processus Décisionnels de Markov

Complexité

Résoudre de manière optimale un DEC-POMDP (ou un DEC-MDP) consiste à trouver une politique optimale jointe $\pi = \langle \pi_1, \cdots, \pi_n \rangle$ composée d'un ensemble de politiques locales π_i. Une politique optimale jointe est une politique qui maximise l'utilité espérée du système. Elle fournit aux agents un comportement optimal, c'est-à-dire qu'à chaque étape de décision, chaque agent choisit l'action susceptible d'augmenter le plus le gain du système.

En raison du manque d'observabilité de l'état du système et des actions des autres agents, la résolution optimale de problèmes de contrôle décentralisé est beaucoup plus difficile que dans le cadre du contrôle centralisé. En effet, les performances du système dépendent de son état global et de l'action jointe exécutée. Cependant, lorsqu'un agent décide quelle action réaliser, il n'a qu'une connaissance partielle de ces informations. Bien que la résolution optimale des POMDPs soit un problème PSPACE-complet, la résolution des DEC-POMDPs est bien plus complexe. Bernstein et al. [Bernstein *et al.*, 2002] ont démontré que résoudre de façon optimale un DEC-POMDP à 2 agents ou plus est un problème NEXP. Il en est de même pour les DEC-MDPs : la résolution optimale d'un MDP est un problème P-complet alors que celle d'un DEC-MDP est NEXP pour 3 agents ou plus.

Lorsque l'état du système est collectivement totalement ou partiellement observable, les agents n'ont pas accès individuellement à l'état global du système et doivent baser leurs décisions sur leurs observations locales. La taille de la politique de chaque agent est alors exponentielle et la complexité du problème est NEXP. Il en est de même pour les DEC-MDPs. Si l'état du système est individuellement totalement observable, c'est-à-dire que chaque agent connaît l'état global du système, alors le problème peut se réduire à un MDP dont la résolution est P-Complet.

Enfin, si l'état global du système n'est pas observable, le problème se réduit à un MDP non observable (NOMDP) connu pour être NP-complet [Papadimitriou et Tsitsiklis, 1987]. Le tableau 3.2 résume l'influence de l'observabilité de l'état global du système sur la complexité du problème.

Observabilité de l' état du système	Individuellement Observable	Collectivement Observable	Collectivement Partiellement Observable	Non Observable
Complexité	P-complet	NEXP-Complet	NEXP-Complet	NP-Complet

TAB. 3.2 – Observabilité et Complexité en temps des DEC-POMDPs et MTDPs

3.4.2 Gestion de la communication

Dans certains systèmes multi-agents, il est possible que les agents communiquent entre eux lors de l'exécution des tâches. Cette communication peut alors leur permettre d'échanger des informations sur leurs observations, leur état local, etc. Ainsi, les agents augmentent leurs connaissances sur les autres agents et sur l'état global du système. Ils peuvent alors se synchroniser plus facilement.

Nous avons décrit au premier chapitre trois types de communication : directe, indirecte, par observation commune de caractéristiques de l'environnement. Nous nous intéresserons, dans ce chapitre, à l'influence de la communication sur l'observabilité des agents. Nous ne traiterons donc que de la communication directe. En effet, cette dernière constitue le seul type de communication qui permette d'atteindre l'observabilité totale lorsqu'il n'existe pas de caractéristiques incontrôlables communément observables par tous les agents [Goldman et Zilberstein, 2004].

Toute communication a généralement un coût dû aux ressources qu'elle consomme (énergie, bande passante, ...) ou bien dû au risque de révéler des informations à d'éventuels agents compétitifs. Lorsqu'un agent a la possibilité de communiquer, il doit tenir compte de l'utilité des informations apportées par la communication et du coût de cette dernière. La décision de communiquer ou non résulte alors d'un compromis entre cette utilité et ce coût. Nous désignerons par communication gratuite une communication directe instantanée dont le coût est nul pour les agents.

Plusieurs formalismes ont proposé d'étendre les DEC-POMDPs afin de permettre la modélisation de la communication entre les agents durant la phase d'exécution des tâches. A chaque étape de décision, un agent peut décider de communiquer ou non. Ensuite, il détermine quelle action réaliser. Comme le montre la figure 3.4, chaque étape de décision est donc décomposée en deux phases : la phase de communication et la phase d'exécution d'une action standard. On désigne par « action standard » toute action ne consistant pas à communiquer des informations avec les autres agents.

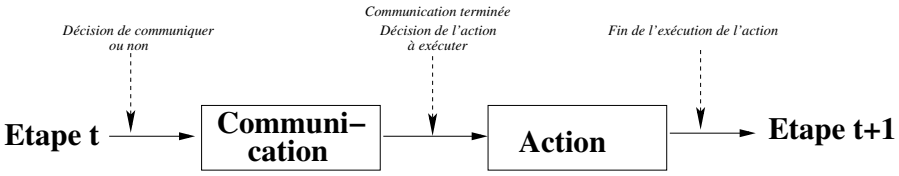


FIG. 3.4 – Différentes phases d'une étape de décision

La politique π_i de chaque agent est alors composée de deux politiques distinctes : une politique de communication π_i^{Σ} et une politique d'exécution des actions standards π_i^{a}.

Goldman et Zilberstein [Goldman et Zilberstein, 2003] ont décrit une extension des DEC-POMDPs, permettant de modéliser la communication entre les agents. Les DEC-POMDP-COM (Processus Décisionnels de Markov Décentralisé Partiellement Observables avec Communication) sont définis de la même manière que les DEC-POMDPs. Deux nouvelles composantes sont cependant ajoutées au formalisme : un langage de communication Σ et une fonction de coût d'envoi des messages C_{Σ}.

Définition 15 ***DEC-POMDP avec Communication***

Un DEC-POMDP-COM est défini par un tuple $< \mathcal{S}, \mathcal{A}, \Sigma, C_{\Sigma}, \mathcal{P}, \Omega, \mathcal{O}, \mathcal{R} >$ *tel que :*

- $\mathcal{S}$ *désigne l'ensemble des états du système.*
- $\mathcal{A} = \langle \mathcal{A}_1, \cdots, \mathcal{A}_n \rangle$ *est l'ensemble des actions jointes et* $\mathcal{A}_i$ *définit l'ensemble des actions* a_i *de l'agent* $\mathcal{A}g_i$.
- Σ *désigne l'alphabet des messages.* $\sigma_i \in \Sigma$ *correspond à un message de l'agent* $\mathcal{A}g_i$.
- C_{Σ} *associe à tout message un coût. Cette fonction est telle que* $C_{\Sigma} : \Sigma \rightarrow \mathbb{R}$.
- $\mathcal{P} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ *définit la fonction de transition.* $\mathcal{P}(s, a, s')$ *correspond à la probabilité que le système passe d'un état* s *à un état* s' *lorsque l'action jointe* a *est exécutée.*
- $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$ *est l'ensemble des observations des agents tel que* Ω_i *est l'ensemble des observations de l'agent* $\mathcal{A}g_i$.

- $\mathcal{O} = \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0,1]$ *définit la fonction d'observation.* $\mathcal{O}(s, a, s', o = \langle o_1, \cdots, o_n \rangle)$ *correspond à la probabilité que chaque agent* $\mathcal{A}g_i$ *observe* o_i *lorsque les agents exécutent l'action jointe* a *à partir de l'état* s *et que le système arrive dans l'état* s'.
- $\mathcal{R}$ *définit la fonction de récompense.* $\mathcal{R}(s, \langle a_1, \cdots, a_n \rangle, s')$ *est la récompense obtenue par le système lorsque les agents exécutent l'action* $\langle a_1, \cdots, a_n \rangle$ *à partir de l'état* $\langle s_1, \cdots, s_n \rangle$ *et arrivent dans l'état* $\langle s'_1, \cdots, s'_n \rangle$.

Remarquons qu'il est également possible d'ajouter la communication au formalisme des DEC-MDPs comme l'ont fait Xuan et Lesser [Xuan *et al.*, 2001].

Pynadath et Tambe [Pynadath et Tambe, 2002] ont également décrit une extension des MTDPs permettant la modélisation de la communication. Un COM-MTDP (Communicative Multiagent Team Decision Problem) est défini par un tuple $< \mathcal{S}, \mathcal{A}, \Sigma, \mathcal{P}, \Omega, \mathcal{O}, B, \mathcal{R} >$. La fonction de récompense $\mathcal{R}$ est étendue afin de modéliser le coût des envois de messages. Ce formalisme est plus général que les DEC-POMDP-COM car il permet, tout comme les MTDPs, de représenter l'état mental des agents à l'aide des états de croyances.

La figure 3.5 présente les relations entre les différents formalismes qui viennent d'être présentés. Notons que les DEC-MDPs appartiennent à l'ensemble des DEC-MDP-COM qui sont un cas particulier des DEC-POMDP-COM. En effet, les DEC-MDPs constituent des DEC-MDP-COM dans lesquels l'alphabet des messages est vide. De plus, les DEC-MDPs constituent un sous-ensemble des DEC-POMDPs : un DEC-MDP est un DEC-POMDP dans lequel les agents ont une observabilité collective totale. Les DEC-POMDPs ne consituent cependant pas une sous-classe des DEC-MDP-COM. Plus formellement : DEC-MDP $\subset$ DEC-MDP-COM $\subset$ DEC-POMDP-COM et DEC-MDP $\subset$ DEC-POMDP mais DEC-POMDP $\not\subset$ DEC-MDP-COM et DEC-POMDP $\subset$ DEC-POMDP-COM.

Le coût de la communication a une influence importante sur la complexité du problème. Lorsque la communication est gratuite, il est possible pour chaque agent de communiquer toutes ses connaissances à tous les autres agents. Si le système est collectivement totalement observable, la communication permet de se ramener à un processus de décision décentralisé complètement observable. Ce dernier peut être représenté par un MMDP dont la résolution est P-complète [Papadimitriou et Tsitsiklis, 1987]. Dans le cas où l'état n'est pas collectivement totalement observable, communiquer permet de ramener le problème à un POMDP dont la résolution est PSPACE [Papadimitriou et Tsitsiklis, 1987]. Si les agents n'ont aucune observabilité, alors le problème correspond à un NOMDP (Processus Décisionnel de Markov Non Observable) qui est

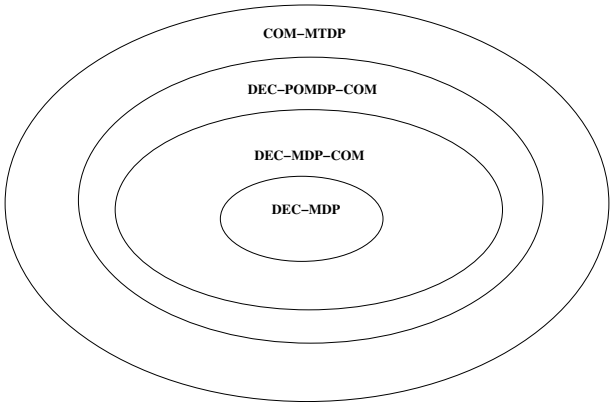


FIG. 3.5 – Relations entre les différents types de Processus Décisionnels de Markov avec Communication

NP-Complet.

Lorsque la communication n'est pas gratuite, chaque agent doit trouver un compromis entre le coût de la communication et les bénéfices résultant des informations qu'il aura obtenues. Résoudre de manière optimale de tels problèmes consiste à trouver une politique optimale jointe $\pi = \langle \pi_1, \cdots, \pi_n \rangle$ où chaque politique locale comprend deux composantes. Chaque agent doit donc construire une politique $\pi_i = \langle \pi_i^{\Sigma}, \pi_i^a \rangle$ telle que π_i^{Σ} et π_i^a maximisent la récompense jointe.

L'ajout de communication non gratuite ne diminue pas la complexité du problème. Le calcul de la politique de communication est aussi difficile que le calcul de la politique d'un agent dans un DEC-POMDP. L'ensemble des actions d'un DEC-POMDP est composé des actions « communiquer » et des actions standards. La politique π_i d'un agent $\mathcal{A}g_i$ est définie et calculée de la même façon que pour les DEC-POMDPs.Un DEC-POMDP-COM a donc la même complexité qu'un DEC-POMDP. De même, les complexités des DEC-MDP-COM et des DEC-MDPs sont identiques.

Lorsque l'état du système est individuellement totalement observable, les agents n'ont aucun besoin de communiquer. Chaque agent peut à partir de ses observations, calculer sa politique. Le problème se ramène en effet, comme dans le cas de la communication gratuite, à un MMDP. Ce MMDP est alors résolu par chaque agent qui en déduit sa politique et celles des autres agents. Dans le cas où les agents n'ont aucune observabilité de l'état du système ni de leur propre état (agents aveugles), le problème se ramène à un NOMDP (NP-complet) [Pynadath et Tambe, 2002]. Ces résultats sont résumés par le tableau 3.3.

Le formalisme des COM-MTDP ne pose aucune hypothèse quant à l'observabilité collective de l'état du système et au coût de la communication. Suivant l'observabilité du système, leur complexité varie.

Observabilité de l' état du système	Individuellement Observable	Collectivement Observable	Collectivement Partiellement Observable	Non Observable
Communication gratuite	P-complet	P-Complet	PSPACE- Complet	NP-Complet
Communication avec un coût	P-complet	NEXP-Complet	NEXP-Complet	NP-Complet

TAB. 3.3 – Observabilité et Complexité en temps des DEC-POMDP-COM et COM-MTDPs

3.5 Propriétés et classes particulières de DEC-POMDPs

Nous avons vu que la résolution des DEC-POMDPs était un problème difficile. Cependant, certaines propriétés, autres que le degré d'observabilité, permettent de diminuer la complexité du problème.

3.5.1 Transitions, observations et buts

Parmi ces propriétés, on peut citer la transition-indépendance, l'observation-indépendance et l'existence d'un but commun à tous les agents. Dans cette section, nous allons définir ces propriétés et étudier leur influence sur la complexité des DEC-POMDPs et des DEC-MDPs. Ces résultats sont également applicables aux MTDPs.

Transition et Observation Indépendantes

La transition-indépendance permet d'identifier des problèmes où les actions des agents sont indépendantes les unes des autres.

Définition 16 ***Transitions Indépendantes***

Un DEC-POMDP est dit à transitions indépendantes si l'espace d'états $\mathcal{S}$ peut être factorisé en n composantes $\mathcal{S}_1$ à $\mathcal{S}_n$ et que la fonction de transition peut être décomposée en un produit de probabilité : $\mathcal{P} = \prod_{i=1}^{n} \mathcal{P}_i$ où $\mathcal{P}_i = Pr(s'_i|s_i, a_i)$.

Pour un DEC-POMDP à deux agents, la propriété de transition-indépendance se traduit par la formule suivante :

$$\forall s_1, s'_1 \in \mathcal{S}_1, \forall s_2, s'_2 \in \mathcal{S}_2, \forall a_1 \in \mathcal{A}_1, \forall a_2 \in \mathcal{A}_2,$$

$$Pr(s'_1|(s_1, s_2), a_1, a_2, s'_2) = Pr(s'_1|s_1, a_1) \wedge$$

$$Pr(s'_2|(s_1, s_2), a_2, a_1, s'_1) = Pr(s'_2|s_2, a_2)$$

Lorsqu'un DEC-POMDP est à transitions indépendantes, la probabilité qu'un agent $\mathcal{A}g_i$ passe d'un état s_i à un état s'_i ne dépend que de l'action a_i qu'il a exécutée. Les actions des autres agents n'ont pas d'influence sur la transition de l'agent $\mathcal{A}g_i$.

L'observation-indépendance caractérise les problèmes où les observations de chaque agent sont indépendantes des actions des autres agents.

Définition 17 ***Observations Indépendantes***

Un DEC-POMDP est dit à observations indépendantes si l'espace d'états $\mathcal{S}$ peut être factorisé en n composantes $\mathcal{S}_1$ à $\mathcal{S}_n$ et que la probabilité d'observation $\mathcal{O}$ peut être décomposée en n probabilités d'observations $\mathcal{O}_i$ telles que :

$$\mathcal{O}_i = Pr(o_i|\langle s_1, \cdots, s_n\rangle, \langle a_1, \cdots, a_n\rangle, \langle s'_1, \cdots, s'_n\rangle, \langle o_1, \cdots, o_{i-1}, o_{i+1}, \cdots, o_n\rangle)$$

Pour un DEC-POMDP à deux agents, la propriété d'observation-indépendance se traduit par la formule suivante :

$$\forall o_1 \in \Omega_1, \forall o_2 \in \Omega_2, \forall s = (s_1, s_2), s' = (s'_1, s'_2) \in \mathcal{S}, \forall a_1 \in \mathcal{A}_1, \forall a_{\in}\mathcal{A}_2,$$

$$Pr(o_1|(s_1, s_2), a_1, a_2, (s'_1, s'_2), o_2) = Pr(o_1|s_1, a_1, s'_1) \wedge$$

$$Pr(o_2|(s_1, s_2), a_1, a_2, (s'_1, s'_2), o_1) = Pr(o_2|s_2, a_2, s'_2)$$

$$\mathcal{O}(o_1, o_2|(s_1, s_2), a_1, a_2, (s'_1, s'_2)) =$$

$$Pr(o_1|(s_1, s_2), a_1, a_2, (s'_1, s'_2), o_2) \times Pr(o_2|(s_1, s_2), a_1, a_2, (s'_1, s'_2), o_1)$$

Un DEC-POMDP à transitions et observations indépendantes modélise un problème où l'influence des actions d'un agent sur les autres, est restreinte. En effet, seule la fonction de récompense dépend de l'action jointe exécutée. Les dépendances entre les agents sont donc formalisées par la fonction de récompense. La conjugaison de ces deux propriétés permet, dans certains cas, de diminuer la complexité.

Lors de la résolution de problèmes de décision décentralisée, les agents doivent faire face au manque d'observabilité de l'état du système. Afin de calculer une politique globale optimale, chaque agent doit garder en mémoire toute la séquence de ses observations. La taille de la politique de l'agent est donc exponentielle en $|\Omega_i|^T$ et son évaluation est exponentielle. De plus, il existe $|\mathcal{A}_i|^{|\Omega_i|^T}$ politiques devant être évaluées pour chaque agent, d'où une complexité NEXP.

Goldman et Zilberstein [Goldman et Zilberstein, 2004] ont montré que la résolution d'un DEC-MDP à transitions et observations indépendantes nécessite que l'agent ne garde en mémoire que

sa dernière observation. Ainsi, la taille des politiques locales des agents diminue. La politique d'un agent $\mathcal{A}g_i$ a alors une taille polynomiale en $|\mathcal{S}_i| \times T$ et son évaluation peut être réalisée en temps polynomial. Cependant, il existe un nombre exponentiel $|\mathcal{A}_i|^{|\mathcal{S}_i| \times T}$ de politiques. Afin de calculer la politique optimale, il est nécessaire d'évaluer tout l'espace de politiques, c'est pourquoi la résolution d'un DEC-MDP à transitions et observations indépendantes est un problème NP-complet.

En ce qui concerne les DEC-POMDPs, les propriétés d'indépendance des transitions et des observations n'entraînent pas de diminution de la complexité. En effet, chaque agent doit, dans tous les cas, mémoriser toute la séquence de ses observations et la taille des politiques locales reste inchangée.

Agents orientés vers un but

Un processus orienté vers un but est un processus dans lequel les agents cherchent à atteindre des états spécifiques du système correspondant à des états où le but a été atteint. Par exemple, des agents devant transférer des objets d'un point A à un point B auront atteint un état but quand tous les objets seront au point B.

Définition 18 ***Orientation par des buts***

Un DEC-POMDP est dit orienté par des buts si les conditions suivantes sont remplies :

1. *Il existe un sous ensemble non vide $\mathcal{G}$ de l'ensemble des états $\mathcal{S}$ représentant les états buts du système. Au moins un état $g \in \mathcal{G}$ est accessible par une politique jointe.*
2. *Le problème a un horizon fini $\mathcal{H}$.*
3. *Toute action a_i d'un agent $\mathcal{A}g_i$ a un coût $C(a_i) < 0$.*
4. *La fonction de récompense est telle que $\mathcal{R}(s, \langle a_1, \cdots, a_n \rangle, s') = \sum_{i=1}^{n} C(a_i)$.*
5. *Si à l'instant $\mathcal{H}$ ($\mathcal{H}$ étant l'horizon du problème), le système a atteint un état but $s \in \mathcal{G}$ alors, une récompense supplémentaire est attribuée au système pour avoir atteint un état but.*

Un tel DEC-POMDP est communément noté GO-DEC-POMDP.

Définition 19 ***Goal Oriented DEC-POMDP à coût uniforme***

Un GO-DEC-POMDP est dit à coût uniforme si toutes les actions ont le même coût.

La résolution de DEC-MDPs à transitions et observations indépendantes, ayant un seul but et un coût uniforme, est P-complet. Chaque agent doit, dans ce type de problème, suivre une politique qui minimise les coûts. En raison de l'hypothèse d'uniformité des coûts, ceci revient à calculer la politique de plus court chemin, par un algorithme de programmation dynamique P-complet.

En revanche la résolution d'un DEC-MDP ou d'un DEC-POMDP orienté par un but, à transitions et observations non indépendantes reste un problème NEXP. La complexité d'un DEC-POMDP orienté par un but et à transitions et observations indépendantes, reste un problème ouvert.

3.5.2 DEC-MDPs dirigés par les événements

Becker et al. [Becker *et al.*, 2004a] ont identifié une autre sclasse de DEC-MDPs dont la complexité est moins importante que celles des DEC-MDPs classiques : les DEC-MDPs dirigés par les événements ou Event Driven Decentralized Markov Decision Processes (ED-DEC-MDP).

Les ED-DEC-MDPs décrivent des problèmes dans lesquels les interactions entre les agents correspondent à des dépendances entre leurs actions. Ces dernières se traduisent par des contraintes entre les agents, comme par exemple des contraintes temporelles ou bien des contraintes d'ordre entre les actions. Cette classe de DEC-MDPs permet de représenter des contraintes sur l'exécution des tâches qui sont souvent rencontrées en pratique et non modélisées dans le formalisme classique des processus décisionnels de Markov décentralisés. Il est en effet possible de formaliser des problèmes dans lesquels le résultat de l'action d'un agent dépend de l'exécution d'autres actions par d'autres agents.

Différents types de relations peuvent ainsi être modélisées. Parmi elles, on trouve les relations de précédence : un agent $\mathcal{A}g_i$ ne peut exécuter la tâche t_i que si l'agent $\mathcal{A}g_j$ a fini d'exécuter la tâche t_j. De même, il est possible de représenter des dépendances de qualité : si l'agent $\mathcal{A}g_j$ a terminé la tâche t_j lorsque la tâche t_i est exécutée, alors la qualité d'exécution de t_i sera meilleure.

Afin de représenter ces dépendances, le langage TAEMS [Decker et Lesser, 1993a] de description des tâches est utilisé. Il permet de décrire la structure des tâches et leurs relations.

Les ED-DEC-MDPs supposent que les états locaux des agents sont localement totalement observables. Enfin, la fonction de récompense $\mathcal{R}$ d'un ED-DEC-MDP est à récompenses indépendantes c'est-à-dire qu'elle peut être décomposée en une somme de fonctions de récompense locales telles que $\mathcal{R}(\langle s_1, \cdots, s_n\rangle, \langle a_1, \cdots, a_n\rangle, \langle s'_1, \cdots, s'_n\rangle) = \sum_{i=1}^{n} R_i(s_i, a_i, s'_i)$.

La résolution d'un ED-DEC-MDP est un problème NP-complet. En effet, un ED-DEC-MDP a une complexité exponentielle en espace d'états et doublement exponentielle en nombre de dépendances. Les ED-DEC-MDPs étant localement totalement observables, une politique fait correspondre une action, à chaque état (et non à chaque observation). La taille d'une telle politique est alors exponentielle en espace d'états. Le nombre d'états étant exponentiel en nombre de dépendances, le nombre de politiques est doublement exponentiel en nombre de dépendances.

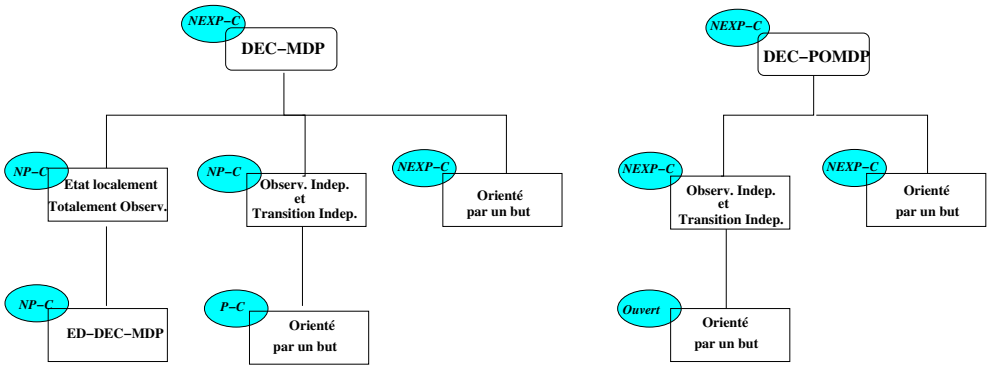


FIG. 3.6 – Complexité des problèmes des classes de DEC-MDP et DEC-POMDP

Les propriétés que nous venons de décrire permettent d'identifier des sous-classes de DEC-MDPs de complexité moins importante. Ces résultats sont résumés par la figure 3.6. La solution optimale à ces problèmes peut alors être obtenue plus facilement. La section suivante présente différents algorithmes permettant de résoudre des DEC-POMDPs généraux ou bien seulement certaines classes de DEC-POMDPs.

3.6 Méthodes de résolution des DEC-POMDPs

Au cours de ces dernières années, plusieurs travaux ont porté sur la résolution des DEC-POMDPs. Les méthodes proposées peuvent être divisées en deux catégories distinctes. La première regroupe les approches dites de résolution exacte, c'est-à-dire cherchant une solution optimale au problème. Nous montrerons que ces méthodes sont très rapidement confrontées à des problèmes de complexité. Un second type d'approches consiste à développer des algorithmes d'approximation de la solution optimale. Ces derniers calculent une solution presque optimale ou optimale localement. La complexité de tels algorithmes est généralement plus raisonnable mais l'optimalité n'est alors plus garantie.

3.6.1 Algorithmes de résolution exacte

Parmi les approches cherchant une solution optimale, nous identifierons deux types de méthodes. Nous présenterons tout d'abord des travaux s'intéressant à la résolution de DEC-POMDPs généraux, sans aucune propriété particulière. Nous décrirons ensuite une approche identifiant des sous-classes de DEC-POMDPs et exploitant les propriétés des problèmes abordés, lors de leur résolution.

Elimination itérative des politiques dominées

Hansen et al. [Hansen *et al.*, 2004] ont décrit un algorithme permettant de résoudre de manière optimale des DEC-POMDPs à horizon fini $\mathcal{H}$. Cet algorithme utilise la programmation dynamique ainsi que des principes de la théorie des jeux tels que la représentation de jeux sous forme normale et l'élimination itérative de stratégies dominées.

Un DEC-POMDP est représenté par un jeu sous forme normale construit récursivement : la forme normale du jeu à l'étape $t+1$ est déterminée à partir de sa forme normale à l'étape t. Chaque étape de construction de la solution est divisée en deux phases : la construction de la forme normale pour l'étape de décision $t+1$ et l'élimination des stratégies faiblement dominées. Cette dernière phase permet d'élaguer l'espace des politiques et ainsi de limiter le nombre de politiques à évaluer.

Dans le cas des DEC-POMDPs à horizon fini, cet algorithme permet d'obtenir une solution optimale. Si l'horizon est infini, nous aboutissons à un ensemble d'équilibres.

Les expérimentations montrent que l'algorithme est beaucoup plus efficace que l'évaluation de toutes les politiques en utilisant la force brute (recherche exhaustive). Cependant, le nombre de politiques à évaluer augmente fortement avec l'horizon ce qui entraîne rapidement une limitation de la taille des problèmes traités. Ainsi, Hansen et al. arrivent à résoudre au maximum jusqu'à l'horizon 4 (4 décisions par agent), des problèmes à 2 agents, 4 états, 2 actions par agent et 2 observations par agent.

Multi-agent A* : MAA*

Szer et al. [Szer *et al.*, 2005] ont proposé une extension multi-agent de l'algorithme de recherche heuristique A* : MAA*. Cet algorithme permet de calculer la politique optimale d'un DEC-POMDP à horizon fini ou infini.

Tout comme l'algorithme classique A*, MAA* procède à une construction incrémentale de la solution. Grâce au principe d'optimalité de Bellman, il est possible de calculer une solution optimale jusqu'à un horizon $t < \mathcal{H}$. Une solution partielle au problème est ainsi obtenue. Une heuristique permet ensuite d'évaluer quelle solution paraît la plus prometteuse pour étendre cette solution partielle : on cherche à développer la solution qui conduira au comportement le plus prometteur entre t et $\mathcal{H}$.

Plusieurs heuristiques ont été décrites afin de calculer rapidement une approximation des politiques entre t et $\mathcal{H}$, et ainsi déterminer quelle solution doit être développée en priorité. Il est proposé de réduire le DEC-POMDP à un MDP ou à un POMDP centralisé pouvant être résolu plus facilement. L'efficacité de l'algorithme est étroitement liée à l'heuristique utilisée. Plus l'heuristique sera précise, plus la solution sera trouvée rapidement. Dans le pire cas, la complexité de cet algorithme est équivalente à celle de la recherche exhaustive.

Les résultats expérimentaux montrent que cet algorithme est plus efficace que l'élimination itérative des politiques dominées proposée par Hansen et al. puisque MAA* stocke moins de politiques en mémoire. Ainsi, dans des problèmes où Hansen et al. [Hansen *et al.*, 2004] peuvent aller jusqu'à l'horizon 4, MAA* calcule une solution jusqu'à l'horizon 5. Au delà, MAA* se trouve également confronté à une explosion de la mémoire nécessaire à la résolution du problème.

La complexité des DEC-POMDPs et des DEC-MDPs généraux est bien trop importante pour songer à résoudre ces problèmes de façon optimale. Les approches que nous venons de présenter montrent bien les limitations de tels algorithmes. En théorie, la solution optimale peut être obtenue. En pratique, le temps et la mémoire requis pour calculer l'optimal sont beaucoup trop importants. Comme nous le montrerons par la suite, limiter la mémoire ou le temps consommés revient alors à sacrifier l'optimalité globale.

Les approches précédentes s'intéressent au problème général des DEC-POMDPs dont la complexité est NEXP. Cependant, nous avons vu précédemment que certaines propriétés du problème permettaient de diminuer sa complexité. Il peut alors être plus facile de calculer la solution optimale.

Résolution de sous-classes de DEC-POMDPs

A partir des travaux de Goldman et Zilberstein sur l'identification de classes de DEC-POMDPs de complexité moins importante, Becker et al. ont proposé un algorithme permettant de résoudre certaines de ces classes de problèmes. CSA (Coverage Set Algorithm) permet de résoudre des DEC-MDPs à transitions et observations indépendantes [Becker *et al.*, 2003, Becker *et al.*, 2004b], mais également des DEC-MDPs dirigés par les événements [Becker *et al.*, 2004a].

Bien que CSA soit présenté dans le cadre de problèmes à deux agents, il est en mesure de traiter des problèmes à n agents ($n \geq 2$). CSA se décompose entre trois étapes : la création de MDPs augmentés, la recherche d'un ensemble optimal de couverture et la recherche d'une solution. Étant donnés deux agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$, un MDP augmenté représente un problème de décision dans lequel un agent $\mathcal{A}g_i$ doit calculer sa politique optimale, sachant une politique fixe de l'agent $\mathcal{A}g_j$.

La première phase de l'algorithme consiste donc à créer, pour chaque politique de $\mathcal{A}g_j$, un MDP augmenté. La deuxième étape réalisée par CSA calcule la politique optimale de chacun de ces MDPs augmentés. Elle permet ainsi d'obtenir l'ensemble des politiques optimales de l'agent $\mathcal{A}g_i$ quelle que soit la politique de l'agent $\mathcal{A}g_j$. L'ensemble de ces politiques est appelé ensemble

de couverture optimal. Définir cet ensemble permet de réduire l'espace de recherche des politiques et d'améliorer les performances par rapport à une recherche exhaustive. Enfin, la troisième et dernière étape, réalisée par CSA, cherche la politique optimale jointe dans l'ensemble de couverture optimal. Pour chaque politique de l'agent $\mathcal{A}g_i$ appartenant à cet ensemble, la politique optimale pour $\mathcal{A}g_j$ est recherchée. La politique jointe résultante est évaluée et la meilleure paire trouvée correspond à la politique optimale jointe.

La complexité de cet algorithme est exponentielle en espace d'états. Dans le cadre de la résolution des ED-DEC-MDPs, la complexité est également doublement exponentielle en nombre de dépendances. Dans le pire cas, CSA a la même complexité que la recherche exhaustive.

Bien que la complexité des sous-classes de problèmes traitées par CSA soit moins importante que celles des DEC-POMDPs généraux, il reste encore très difficile de résoudre des problèmes de tailles conséquentes. Dans le cas des ED-DEC-MDPs, CSA ne peut en pratique résoudre que de petits problèmes à 2 agents et très peu de dépendances. A partir du moment où le nombre d'agents ou de dépendances augmente, le temps nécessaire pour trouver la solution devient beaucoup trop important et la solution ne peut être obtenue.

3.6.2 Algorithmes de résolution approchée

Les approches que nous venons de présenter démontrent les limites de la résolution optimale des DEC-POMDPs. Pour des raisons de complexité, cette dernière n'est pas envisageable si nous souhaitons traiter des problèmes de taille raisonnable. Ainsi, plusieurs travaux se sont intéressés à la mise en place d'algorithmes permettant d'obtenir une approximation de la politique optimale.

Limitation de la mémoire

Afin de pallier aux problèmes d'explosion de mémoire dus au nombre exponentiel de politiques à stocker, Bernstein et al. [Bernstein *et al.*, 2005] ont proposé un algorithme pour la résolution de DEC-POMDPs ne consommant qu'une quantité fixe de mémoire et pouvant traiter des problèmes à horizon fini ou infini. La politique de chaque agent est représentée par un contrôleur stochastique à états finis. Un contrôleur dit de corrélation permet aux agents de coordonner leurs politiques.

L'algorithme fonctionne par amélioration itérative des contrôleurs : à chaque itération un contrôleur est mis à jour de façon à augmenter l'utilité jointe des agents. La convergence vers un optimum global n'est pas garantie. Cependant, il est démontré que la qualité de la solution augmente de façon monotone à chaque itération. L'algorithme atteint par conséquent un optimum local. Les expérimentations montrent que plus le nombre de nœuds de chaque contrôleur augmente, plus la qualité de la solution augmente. En revanche, la quantité de mémoire nécessaire augmente elle aussi.

Chaque itération étant réalisée en un temps polynomial, des problèmes plus grands que ceux

traités par les algorithmes de résolution optimale, peuvent être envisagés. L'augmentation de la taille des problèmes pouvant être résolus est réalisée au détriment de la qualité de la solution : de plus grands problèmes peuvent être traités mais la solution optimale n'est pas trouvée.

La taille des problèmes résolus reste cependant limitée à un petit nombre d'agents puisque chaque itération est exponentielle en fonction du nombre d'agents.

Algorithmes de co-évolution itératifs

Différents algorithmes de co-évolution itératifs ont été proposés afin de permettre l'obtention d'une politique localement optimale. Le principe de ces algorithmes consiste à améliorer la politique d'un agent en fixant les politiques de tous les autres agents. Pour un système à n agents, étant fixées les politiques de $n-1$ agents, l'algorithme calcule la politique optimale de l'agent restant. Ce principe est répété itérativement jusqu'à ce qu'il ne soit plus possible d'améliorer aucune des politiques des agents.

Nair et al. [Nair *et al.*, 2003] ont décrit un algorithme basé sur ce principe de co-évolution : JESP (Joint Equilibrium Based Search for Policies). Ce dernier permet de résoudre des problèmes formalisés sous forme de MTDP et ayant des observations indépendantes.

A chaque itération, la politique d'un agent est modifiée afin d'améliorer la politique jointe. L'algorithme converge vers un optimum local correspondant à un équilibre de Nash. Cet algorithme est en moyenne beaucoup plus rapide que la recherche exhaustive. Dans le pire cas, la complexité est toutefois équivalente à celle de la recherche exhaustive.

Nair et al. ont proposé une amélioration de JESP utilisant les principes de la programmation dynamique : DP-JESP (Dynamic Programming Joint Equilibrium Based Search for Policies). DP-JESP est plus rapide que JESP en raison de l'utilisation du principe d'optimalité pour l'évaluation des politiques. Il peut donc traiter des problèmes plus grands que ceux résolus par JESP. Malgré tout, la taille des problèmes traités aussi bien par JESP que par DP-JESP, reste limitée. Sur le scénario à deux agents du tigre et du trésor, JESP résout le problème jusqu'à l'horizon 3 et DP-JESP atteint l'horizon 7. Ces problèmes restent assez petits par rapport à des problèmes réels. La complexité de ces algorithmes étant exponentielle, on peut en déduire qu'ils ne sont pas en mesure de traiter des problèmes de grande taille.

L'algorithme LID-JESP (Locally Interacting Distributed Joint Equilibrium Based Search for Policies) [Nair *et al.*, 2005] utilise l'algorithme JESP et les principes de l'optimisation distribuée de contraintes pour la résolution de DEC-MDPs à observations et transitions indépendantes, dans lesquels les interactions entre les agents restent locales. LID-JESP exploite la localité des interactions entre les agents afin d'améliorer les performances de JESP. Chaque agent possède

un ensemble de voisins (des agents) avec lesquels il est en interaction. Par définition, tout agent n'appartenant pas au voisinage d'un agent $\mathcal{A}g_i$ n'est pas influencé par les actions de $\mathcal{A}g_i$. La politique de l'agent $\mathcal{A}g_i$ n'influence donc que ses voisins. LID-JESP tire parti de cette topologie des interactions afin de calculer de manière distribuée les politiques des différents voisinages. Les politiques des agents d'un même voisinage sont calculées par un algorithme de co-évolution itératif identique à JESP. LID-JESP permet ainsi un gain de temps dans la résolution de problèmes où chaque agent n'interagit qu'avec un nombre restreint d'autres agents.

Chadès et al. [Chadès *et al.*, 2002] ont également décrit un algorithme de co-évolution itératif permettant de résoudre des DEC-POMDPs. Contrairement à JESP, aucune contrainte n'est imposée concernant l'indépendance des observations.

Chadès et al. ont tout d'abord proposé deux algorithmes de co-évolution permettant de résoudre les MMDP. Ces algorithmes aboutissent à un optimum local équivalent à un équilibre de Nash.

Une adaptation de ces algorithmes aux DEC-POMDPs a ensuite été proposée. Afin de rendre compte de l'observabilité partielle de l'état du système, chaque agent modélise le problème à l'aide d'un MDP-Subjectif. Un MDP-Subjectif est un MDP dans lequel les états sont remplacés par les perceptions locales de l'agent. Ce formalisme est équivalent à un POMDP sans états de croyances, ni historique, et dans lequel on travaillerait directement sur les observations locales des agents. A chaque itération de l'algorithme, un agent, choisi au hasard, reçoit les politiques locales des autres agents. Ces dernières étant supposées fixes, l'agent peut alors construire son MDP-Subjectif et ensuite le résoudre afin de déterminer sa politique optimale.

Un MDP-Subjectif n'étant pas markovien, sa résolution ne permet pas d'obtenir une politique optimale. Dans le cas du contrôle décentralisé, l'algorithme de co-évolution proposé par Chadès et al. ne garantit donc pas la convergence vers un équilibre de Nash. La qualité des politiques obtenues n'est garantie que dans le cas du contrôle centralisé ou de l'observabilité totale (on peut alors formaliser le problème par un MMDP). Comme nous l'avons remarqué précédemment, ces hypothèses sont rarement vérifiées dans les systèmes multi-agents. Par ailleurs, les résultats expérimentaux montrent que ces algorithmes ne peuvent pas traiter des problèmes à fortes dépendances entre les agents.

Apprentissage par descente de gradient

L'algorithme d'apprentissage par descente de gradient proposé par Peshkin et al. [Peshkin *et al.*, 2000] retourne également un optimum local. Cet algorithme d'apprentissage distribué permet aux agents de déterminer individuellement quelle politique exécuter afin de maximiser la

récompense globale.

Chaque politique est représentée par un contrôleur à états finis. A partir d'un sous-ensemble de politiques jointes, un optimum local est déterminé. Contrairement à JESP, cet optimum peut ne pas être un équilibre de Nash.

Malgré la décentralisation de l'apprentissage, les applications de cet algorithme restent limitées à des problèmes de petite taille.

Jeux Bayésiens

Emery-Montemerlo et al. [Emery-Montemerlo *et al.*, 2004] ont également proposé un algorithme décentralisé pour la résolution de DEC-POMDPs. Cette approche consiste à approximer un DEC-POMDP par une série de jeux bayésiens. La politique du DEC-POMDP est obtenue par concaténation de politiques plus petites résultant de la résolution des jeux bayésiens.

L'algorithme proposé alterne planification et exécution. A chaque étape de planification, un jeu bayésien est résolu par chaque agent afin d'obtenir la politique pour cette étape. Cette politique est ensuite exécutée puis, un nouveau jeu bayésien est résolu, et ainsi de suite.

La résolution d'un jeu bayésien nécessite le recours à une heuristique afin de déterminer la fonction d'utilité. Les performances de cette approche dépendent donc du problème traité et de l'heuristique utilisée.

Par ailleurs, lors de la résolution d'un jeu bayésien, il est nécessaire que chaque agent considère toutes les valeurs possibles des variables qu'il ne peut observer (comme par exemple la position des autres agents). Ceci conduit à une explosion des valeurs à prendre en compte et restreint la taille des problèmes pouvant être considérés.

Approche heuristique

Les travaux utilisant des méthodes heuristiques se sont essentiellement intéressés à la résolution de DEC-POMDP-COM.

Goldman et Zilberstein [Goldman et Zilberstein, 2003] ont présenté une approche de choix glouton pour la résolution du problème de la rencontre d'agents. Deux agents se déplaçant sur une grille doivent se rencontrer le plus tôt possible. Chaque agent ne peut observer la position de l'autre. Les déplacements étant incertains, communiquer permet aux agents d'échanger des informations sur leurs positions respectives et éventuellement réviser leur point de rencontre en cas de déviation. La communication évite ainsi de retarder l'instant de la rencontre entre les agents.

La communication ayant un coût, le problème consiste à déterminer une politique de communication pour chaque agent, lui indiquant quand communiquer. Si la communication est prohibitive, la politique optimale consiste à ne jamais communiquer. Si la communication est gratuite, la

politique optimale est de communiquer tout le temps. Dans les autres cas, la politique optimale se situe entre ces deux extrêmes.

Goldman et Zilberstein ont proposé une méthode permettant de calculer une approximation de la politique de communication optimale. Elle correspond à un choix myope des agents qui ne perçoivent pas leur possibilité de communication future : à chaque étape, chaque agent détermine sa politique de communication en supposant qu'il ne lui reste qu'une seule possibilité de communication. Cette méthode contraint les agents à ne communiquer que s'ils en ont vraiment besoin. Elle permet d'obtenir une approximation de la politique optimale et conduit à de bons résultats dans le problème considéré.

Xuan et al. [Xuan *et al.*, 2001] ont également présenté différentes approches heuristiques pour la résolution du problème de rencontre sous incertitude. Ils ont cependant posé l'hypothèse que les états des agents étaient localement totalement observables.

Aussi bien dans ce travail que dans celui de Goldman et Zilberstein, les heuristiques proposées ne sont évaluées que sur un problème précis. Leurs performances sur d'autres types de problèmes ne sont pas présentées. Ces travaux montrent que la politique de communication optimale se situe quelque part entre les deux politiques extrêmes qui sont « toujours communiquer » et « ne jamais communiquer ». En fonction du problème considéré, l'une ou l'autre des heuristiques va s'avérer la plus appropriée.

Ces travaux suggèrent l'utilité de la mise en place d'algorithmes pouvant résoudre différents types de problèmes. En effet, les méthodes heuristiques sont souvent adaptées à un certain type de problème et ne donnent pas les résultats escomptés lorsque le problème est modifié même légèrement. En raison de la complexité des problèmes, il n'est pas réaliste de chercher une politique optimale, il est donc préférable de se tourner vers la mise en oeuvre d'algorithme permettant d'obtenir des solutions approximant la politique optimale.

3.7 Discussions sur les approches existantes

Nous venons de passer en revue les différentes approches existantes pour la résolution des DEC-POMDPs. Nous avons pu constater la diversité des méthodes employées. Chaque approche posant un certain nombre d'hypothèses sur les problèmes étudiés, les propriétés des DEC-POMDPs considérés varient d'une méthode à l'autre. La figure 3.7 établit une classification de ces méthodes suivant les caractéristiques des problèmes étudiés et le type d'optimalité obtenu.

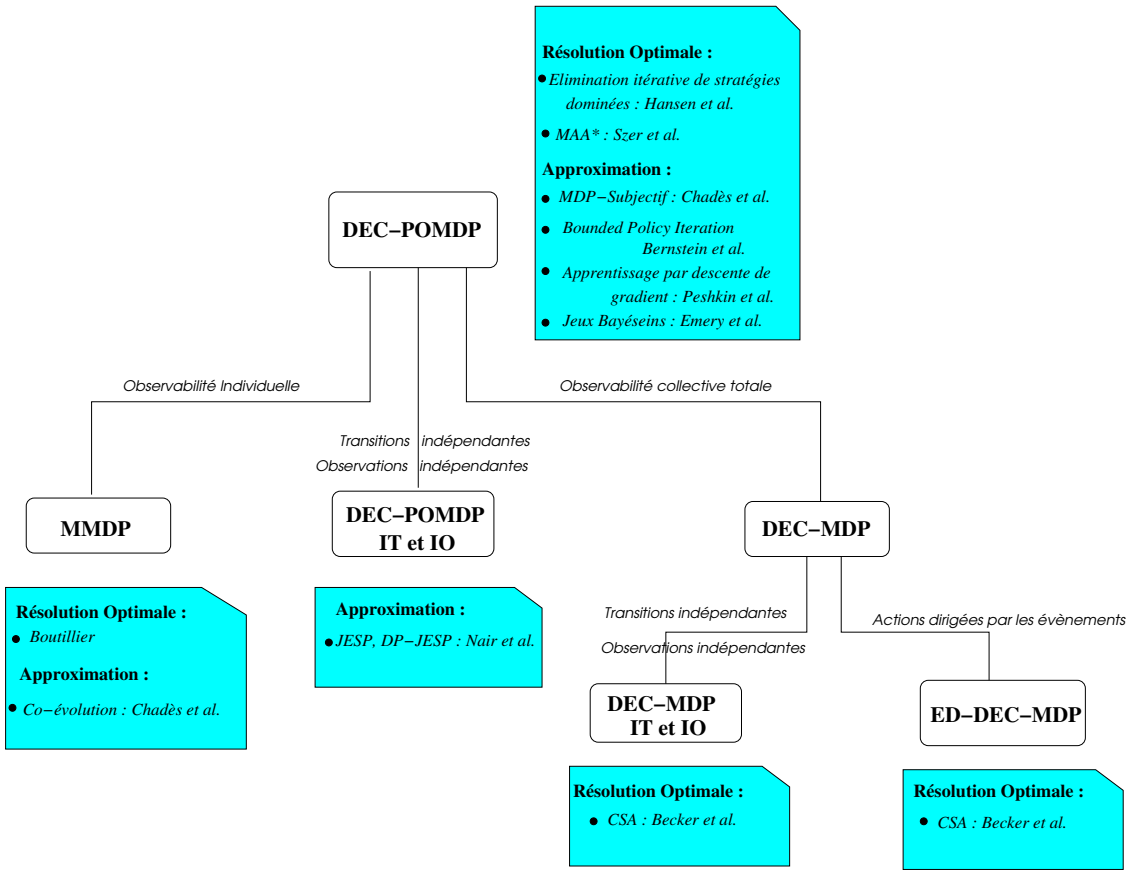


FIG. 3.7 – Classification des approches existantes

3.7.1 Complexité de la résolution optimale

La description des approches existantes nous a tout d'abord permis de mettre en avant la complexité de la résolution des DEC-POMDPs. L'étude des différents algorithmes de calcul d'une politique optimale montre bien que l'obtention d'une telle solution, pour des problèmes de taille raisonnable, n'est pas envisageable. En effet, ces algorithmes se trouvent limités à la résolution de problèmes de taille bien trop petite pour pouvoir envisager traiter des problèmes réels.

L'identification de classes de DEC-POMDPs de complexité moins importante permet de traiter des problèmes un peu plus complexes. Cependant, dans le cas des problèmes NP-complet, comme ceux abordés par Becker et al., la diminution de la complexité ne s'avère pas suffisante. Les problèmes restent difficiles et la complexité exponentielle des algorithmes restreint la taille des problèmes considérés.

3.7.2 Approximation et optimalité

Obtenir une solution optimale pour des problèmes de grande taille apparaît donc peu réaliste. Par ailleurs, l'optimalité est un critère qui peut être moins important qu'il n'y paraît. Dans de nombreux problèmes, on souhaite avant tout obtenir une solution de bonne qualité. L'optimalité passe au second plan. Bien entendu, s'il est possible d'obtenir la solution optimale, cette dernière est préférée. Cependant, quand l'obtention d'une telle solution demande des ressources (en temps, en mémoire, en énergie) non envisageables, une solution presque optimale sera préférée à pas de solution du tout. Dans de telles situations, on cherche à obtenir la meilleure solution possible en fonction du temps et des ressources à notre disposition.

Nous avons pu constater que le principe des méthodes d'approximation résidait dans le fait de sacrifier l'optimalité afin de réduire la complexité des algorithmes et de traiter des problèmes de plus grandes tailles. Certaines approches cherchent des optima locaux, d'autres retournent une solution presque optimale. Les résultats expérimentaux obtenus à partir de ces méthodes montrent que la taille des problèmes traités est plus importante. Nous pouvons donc en déduire que les méthodes d'approximation offrent une voie prometteuse pour la résolution de problèmes réels. Cependant, la taille des problèmes résolus reste encore limitée (pour DP-JESP des résultats expérimentaux sont décrits jusqu'à l'horizon 7). L'utilisation de ces méthodes ne constitue donc pas à elle seule une réponse à la question de l'applicabilité aux problème réels.

3.7.3 Applicabilité

Nous avons jusqu'à présent essentiellement débattu de la taille des problèmes pouvant être résolus par les approches existantes. Cependant, nous avons pu identifier d'autres difficultés pour utiliser ces travaux.

Les systèmes multi-agents présentent en général un certain nombre de contraintes. En ce qui concerne l'exécution des tâches (ou actions), nous avons constaté qu'il était souvent nécessaire de respecter des contraintes temporelles ou des contraintes d'ordre entre les exécutions des tâches. De plus, les agents doivent faire face à des contraintes physiques comme l'impossibilité de communiquer, la limitation de ressources, etc. Toutes ces contraintes ne sont pas prises en compte dans le formalisme des DEC-POMDPs. Il en résulte que les solutions obtenues à partir de la résolution de DEC-POMDPs ne sont pas toujours applicables dans des systèmes réels car elles ne respectent pas les contraintes imposées.

Les ED-DEC-MDPs décrits par Becker et al. ont proposé un début de réponse en permettant la gestion de certaines dépendances entre les agents. Aucune autre approche ne permet de formaliser de telles données du problème. Néanmoins, Becker propose une résolution exacte de ces problèmes ce qui l'applicabilité de son approche à des problèmes de petite taille.

Afin d'offrir un champ d'application plus important à nos travaux, il nous semble donc pri-

mordial de développer une approche permettant de formaliser un certain nombre de contraintes. Les algorithmes de résolution doivent alors être en mesure de tenir compte de ces contraintes lors de la recherche d'une solution.

Conclusion

Dans ce chapitre, nous nous sommes intéressée à la résolution de problèmes décisionnels multi-agents. Ces derniers ont été mis en relation avec la théorie des jeux, en particulier les jeux de Markov. Nous nous sommes ensuite intéressée plus particulièrement aux systèmes multi-agents coopératifs qui constituent le sujet de cette thèse. Nous avons montré l'adéquation des processus décisionnels de Markov décentralisés pour la formalisation des problèmes de décision multi-agents.

L'étude des DEC-POMDPs et de leurs propriétés a montré leur importante complexité algorithmique. Nous avons décrit les différentes approches existantes pour la résolution de DEC-POMDPs. Les approches de résolution exactes et approximées ont été différenciées ce qui nous a permis d'illustrer les limitations du calcul de la solution optimale.

Nous nous sommes alors intéressée à l'applicabilité de ces méthodes dans des systèmes réels. Les approches par approximation nous paraissent plus prometteuses bien qu'elles souffrent elles aussi de certaines limitations. De plus, nous avons pu mettre en avant les déficits des modèles existants quant à la gestion des contraintes du problème. L'applicabilité des travaux existants reste donc limitée à des problèmes académiques. Afin de réduire l'écart entre les problèmes traités par ces modèles et les applications réelles, il nous parait nécessaire de pouvoir modéliser un certain nombre de contraintes. De plus, des algorithmes efficaces doivent être développés afin de résoudre des problèmes de taille conséquente tout en respectant ces contraintes.

Nous tenons à mentionner que l'étude des DEC-POMDPs et des DEC-MDPs constitue un domaine de recherche actif. De ce fait, de nouveaux travaux, non décrits précédemment, sont parus pendant la rédaction de ce document. Ces approches souffrent toutefois du même mal que les approches qui viennent d'être présentées.

Conclusion de la partie I

Cette première partie a été l'occasion de poser les fondements du travail exposé dans ce document. Après avoir décrit la notion d'agent, nous avons motivé le recours à une multiplicité d'entités intelligentes et nous avons présenté les systèmes multi-agents. Nous avons plus particulièrement discuté des difficultés liées à la réalisation d'une modélisation parfaite des lois et phénomènes régissant de tels systèmes. Nous avons donc été conduite à aborder la notion d'incertitude. Nous avons alors choisi de traiter des environnements stochastiques, représentant ainsi les imprécisions de modélisation par des probabilités. Nous avons également discuté des problèmes d'observabilité et de leur influence sur le processus délibératif de chaque agent. Ceci nous a amenée à nous intéresser plus particulièrement au sujet constituant le cœur de notre travail : la **prise de décision multi-agent**.

Le second chapitre a présenté les fondements de la formalisation des problèmes décisionnels dans des environnements stochastiques. Après avoir introduit les chaînes de Markov, nous avons présenté un modèle plus adapté à la prise de décision : les processus décisionnels de Markov. Nous avons montré leur adéquation pour la résolution de problèmes décisionnels mono-agent ce qui nous a conduits à nous intéresser à leur adaptation au domaine multi-agent.

Les processus décisionnels de Markov décentralisés et leurs variantes ont alors été présentés. Nous avons insisté sur l'importante complexité inhérente à la résolution optimale des DEC-MDPs. Afin de surpasser les difficultés de résolution émanant de cette forte complexité, deux approches ont été identifiées : la recherche de caractéristiques diminuant la complexité des problèmes et la recherche d'une solution approchée. Nous avons alors passé en revue différents travaux prenant le parti de l'une ou l'autre de ces approches. Cette étude nous a permis de mettre en évidence les lacunes des approches existantes. Nous avons en effet pu constater que quelle que soit la méthode utilisée, les problèmes résolus restaient de très petite taille restreignant ainsi l'applicabilité de ces travaux. Nous avons alors observé un réel besoin quant au développement d'approches efficaces pouvant être appliquées à des problèmes de grandes tailles.

L'étude de la complexité des DEC-MDPs et des approches proposant une solution optimale, nous a conduit à penser que la recherche de l'optimum n'était pas envisageable. En effet, comme nous l'avons expliqué lors de la présentation des différentes classes de complexité, il est préférable, face à de telles complexités, de s'orienter vers la recherche d'une solution approchée nous fournissant un certain degré de satisfaction. Cette constatation reprend les idées de Simon : étant donné qu'il n'est pas envisageable de considérer toutes les alternatives possibles, il est nécessaire de se limiter à un critère de rationalité limitée.

L'étude menée lors du précédant chapitre, nous a également permis de constater que la recherche d'une approximation ne constituait pas à elle seule une solution pour traiter des problèmes de tailles importantes. Dans le travail qui va suivre, nous proposerons donc de conjuguer les deux approches envisagées par les travaux existants : identifier des caractéristiques du problème permettant une résolution plus efficace et rechercher une approximation de la solution optimale.

Par ailleurs, nous avons pu observer que peu d'approches proposaient de prendre en compte des contraintes sur le problème telles que les contraintes temporelles sur l'exécution des actions. Dans un grand nombre de problèmes réels, omettre ces contraintes conduit à des solutions non viables ne pouvant être utilisées par les agents. Nous avons alors mis en évidence, la nécessité de considérer un certain nombre de contraintes complexes sur l'exécution des actions lors de la résolution des problèmes décisionnels multi-agents.

Cette première partie nous a permis d'établir les axes directeurs de notre travail. Nous allons ainsi nous orienter vers la mise en place d'un formalisme pour la représentation de problèmes de décision séquentielle sous incertitude dans les systèmes-multi-agents. Nous souhaitons que ce formalisme permette la représentation de contraintes diverses telles que les contraintes temporelles sur l'exécution des tâches, les contraintes de ressources, etc. Nous proposerons ensuite un algorithme de résolution des problèmes envisagés. Cet algorithme d'approximation de la politique optimale devra être en mesure de prendre en compte les différentes contraintes imposées et de résoudre des problèmes de tailles importantes[21].

[21] Nous discuterons par la suite de l'ordre de grandeur des problèmes de tailles importantes.

Deuxième partie

Un DEC-MDP pour la gestion de contraintes sous incertitude

Introduction de la partie II

Après avoir introduit dans la partie précédente le contexte général et les concepts clefs de notre travail, nous allons à présent définir plus précisément la problématique de notre thèse.

Comme nous l'avons précédemment mentionné, nous nous intéressons à des systèmes multi-agents coopératifs pour lesquels l'environnement est incertain et partiellement observable. Nous avons par ailleurs présenté différents modèles markoviens permettant la représentation de problèmes décisionnels multi-agents et nous avons discuté de la nécessité de représenter des contraintes complexes et de gérer des problèmes de grande taille. Cette partie va permettre d'établir le lien entre l'état de l'art que nous venons de réaliser, nos travaux et leurs applications. Nous allons ainsi préciser clairement les problématiques auxquelles nous nous intéressons.

A partir de la présentation de différents domaines applicatifs de notre travail, nous mettrons en lumière les propriétés qui caractérisent ces applications et identifierons les problèmes auxquels nous nous attaquons. Nous procéderons à une formalisation des problèmes envisagés puis nous nous intéresserons à la modélisation de ces problèmes sous forme de DEC-MDPs. Nous proposerons alors un modèle adapté à la gestion des contraintes que nous aurons recensées et nous en détaillerons les différentes composantes. Enfin, nous discuterons des propriétés des problèmes traités et de leur complexité.

Chapitre 4

Vers le contrôle de l'exécution d'une mission

Dans les pages qui précèdent, nous avons étudié les différentes approches existantes pour la résolution de processus décisionnels de Markov décentralisés. Nous avons alors montré que les caractéristiques des problèmes considérés différaient d'une approche à une autre. Devant la diversité des problèmes envisagés, il nous semble primordial, avant toute chose, de définir clairement les problèmes que nous traiterons dans cette thèse. Cette présentation fait l'objet du présent chapitre.

Afin de justifier l'ensemble des propriétés et des contraintes associées aux problèmes que nous considérons, nous allons tout d'abord introduire notre contexte applicatif. Nous allons procéder à la description des différentes applications ayant inspiré notre travail. Le point de départ de cette thèse consistait en l'utilisation des processus décisionnels de Markov pour le développement de colonies de robots autonomes. C'est pourquoi, le lecteur pourra constater que toutes les applications envisagées se rattachent au domaine de la robotique.

Le survol de ces applications nous permettra d'identifier les caractéristiques de ces applications et de montrer leurs importantes similitudes. Nous pourrons alors établir une formalisation claire des problèmes et des questions auxquels nous attacherons une attention particulière dans le reste du document.

4.1 Contexte applicatif et motivations

Un robot est un agent devant réaliser un ensemble de tâches en manipulant le monde physique dans lequel il évolue [Russell et Norvig, 2003]. Deux catégories de robots peuvent être identifiées. La première est constituée des robots manipulateurs ancrés physiquement dans un lieu précis et chargés de réaliser une tâche répétitive ou de précision. Ces robots peuvent ainsi être utilisés

FIG. 4.1 – Mars Pathfinder

FIG. 4.2 – Spirit

pour assembler des pièces de voitures, emballer des tablettes de chocolat, visser des bouchons sur des tubes de dentifrices ou bien assister les médecins dans des interventions chirurgicales délicates. Ces robots sont sûrement à l'heure actuelle les plus répandus puisqu'ils sont utilisés par milliers dans l'industrie.

La seconde catégorie de robots, celle à laquelle nous nous intéresserons, est constituée des **robots mobiles** c'est-à-dire se déplaçant dans un environnement à l'aide de roues, de jambes, de pattes, ... En raison de la multitude de leurs applications, il existe une grande diversité de robots mobiles : UAV[22], ULV[23], robots explorateurs (Sojourner, Spirit), robots démineurs, robots humanoïdes (Qrio, Asimo), robots tondeuses ou aspirateurs, ... Parmi la multitude des robots mobiles, nous nous intéresserons plus particulièrement aux robots autonomes évoluant en groupes. Ces **colonies de robots** peuvent, par exemple, être destinées à l'exploration, au sauvetage de victimes ou au déchargement de marchandises.

4.1.1 Robots explorateurs

Nous allons tout d'abord nous intéresser aux robots explorateurs autonomes. De tels robots sont généralement chargés d'intervenir dans des zones hostiles ou difficiles d'accès pour l'homme. Ils peuvent ainsi être envoyés sur des planètes éloignées comme Mars, ou bien être chargés d'explorer des sites minés ou des zones radioactives. Ils ont alors pour mission d'explorer un certain nombre de sites sur lesquels ils doivent réaliser différentes tâches.

Les robots explorateurs les plus célèbres sont sans nul doute les robots Sojourner (1997) et plus récemment les deux MER (Mars Exploration Rovers) de la NASA, Spirit et Opportunity (2003), qui ont été les premiers à « alunir » sur Mars et ont permis d'obtenir de nombreuses

[22] Unmanned Air Vehicle
[23] Unmanned Land Vehicle

informations sur la planète rouge. Ils ont également démontré les potentialités de la robotique exploratoire et ont ainsi ouvert une voie prometteuse vers l'utilisation de robots pour l'exploration d'autres planètes.

Commençons par décrire les caractéristiques d'un robot explorateur sur Mars. Chaque jour, le robot reçoit un ensemble de tâches à réaliser (observations, mesures, déplacements, etc.). Celles-ci sont destinées à accroître les connaissances des scientifiques sur Terre. Ainsi, plus le robot renverra d'informations pertinentes, plus le succès de sa mission sera important. Son critère de performance consiste donc à maximiser la quantité d'informations utiles reçues sur Terre.

La réalisation des tâches doit cependant respecter un certain nombre de contraintes [Cardon *et al.*, 2001, Bresina *et al.*, 2002, Zilberstein *et al.*, 2002] :

- **contraintes temporelles** : Les tâches ne peuvent être réalisées n'importe quand. En effet, leur exécution doit respecter des contraintes temporelles. Par exemple, il est préférable de réaliser les déplacements pendant la journée puisque les robots sont alimentés par des panneaux solaires. Pour des raisons de luminosité, les photographies doivent être prises en début ou en fin de journée, sinon elles seront inexploitables à cause de la trop forte luminosité. De même, les relevés atmosphériques doivent être faits de nuit, au lever ou à la tombée du jour. Les communications nécessitent, quant à elles, que le satellite soit correctement aligné, ce qui n'est le cas que quelques heures par jour.
- **pré-conditions** : Pour être exécutée, chaque tâche requiert qu'un certain nombre de pré-conditions soient satisfaites. Prenons le cas d'un robot chargé de réaliser une mesure de la composition du sol sur un site donné. Pour ce faire, il est nécessaire que le robot se soit rendu sur le site, qu'il ait déployé et calibré ses appareils de mesure et que ces derniers soient correctement positionnés sur la zone à analyser. Si ces conditions ne sont pas remplies lorsque le robot entreprend la mesure, l'exécution de la tâche échouera.
- **contraintes de ressources** : Chaque tâche consomme une certaine quantité de ressources qui doit par conséquent être disponible pour mener à bien l'exécution. Les ressources peuvent prendre différentes formes, il peut s'agir d'énergie mais également de capacité mémoire (pour le stockage de photographies ou des résultats des mesures) ou de bande passante (pour l'envoi de données).

Afin de maximiser leur mesure de performance, les robots explorateurs ne doivent donc pas seulement se contenter d'exécuter leurs tâches, ils doivent également s'assurer du respect des contraintes sur l'exécution des tâches.

Par ailleurs, ces robots évoluent dans des environnements partiellement observables, stochastiques, séquentiels, dynamiques et continus. En raison de la connaissance incomplète de l'environnement, celui-ci est considéré comme incertain. De ce fait, l'exécution d'une tâche peut

conduire à différents résultats. Une tâche peut alors avoir différentes durées possibles et consommer différentes quantités de ressources. Lorsque ces quantités sont continues, l'exécution d'une tâche peut mener à une infinité de situations différentes.

Enfin, les robots doivent faire face à une **limitation des communications** avec la Terre. Dans le cadre des robots martiens, les communications sont réalisées via un satellite en orbite autour de Mars. En raison de sa rotation en orbite, ce satellite n'est disponible que quelques heures par jour. Les communications avec la Terre sont donc restreintes. De plus, toute communication nécessitant des ressources et du temps[24] il est nécessaire de limiter les échanges au strict nécessaire. Les fenêtres de communication sont alors réservées en priorité à l'envoi des données collectées et à la réception de la liste des tâches à réaliser. Le reste de la journée, lorsque le satellite n'est pas correctement aligné, les robots ne peuvent compter sur une quelconque communication. Ils doivent donc agir de manière autonome.

En plus de l'autonomie, les robots doivent être en mesure de développer des processus de décision nécessitant peu de calculs. En raison des contraintes physiques auxquelles sont soumis les engins spatiaux, des composants spécifiques sont développés. Ces derniers sont par exemple capables de résister à de fortes radiations et à de très basses températures. En contrepartie, ils possèdent des capacités limitées par rapport à celles des composants classiques auxquels nous sommes habitués : les processeurs sont plus lents et ne disposent que de faibles quantités de mémoire vive. La puissance de calcul se trouve alors limitée restreignant ainsi la rapidité d'exécution des programmes. De plus, la taille des programmes et la quantité de mémoire requise par leur exécution doivent être minimisées.

L'envoi de robots sur Mars a démontré la faisabilité de tels projets et a permis de tirer des leçons sur la façon de concevoir ces robots. Ainsi Sojourner [Mishkin *et al.*, 1998], le premier robot envoyé sur Mars, a démontré la nécessité de développer des robots qui soient autonomes. En effet, ce robot téléopéré depuis la Terre exécutait une suite de commandes générée et envoyée par les opérateurs de la NASA. Outre l'importance du travail nécessaire pour établir ces suites d'instructions, cette méthode s'est avérée particulièrement inadaptée pour faire face à l'incertitude de l'environnement. Ce robot passait ainsi 40 à 75% de son temps à ne rien faire : se trouvant dans une situation imprévue, il devait alors attendre la prochaine fenêtre de communication avec les opérateurs sur Terre qui lui indiquaient comment agir.

[24] Le temps de communication entre Mars et la Terre oscille entre 4 et 20 minutes en fonction de la position des deux planètes.

La seconde génération de robots [Ai-Chang *et al.*, 2004] a alors été conçue afin d'offrir une plus grande autonomie et s'affranchir d'un commandement à distance. Ainsi, ces robots toujours en activité, reçoivent chaque jour un ensemble de tâches à exécuter et décident eux-mêmes de leur plan d'actions. Cette méthode a permis d'augmenter l'efficacité des robots qui ont déjà pu parcourir plusieurs kilomètres. Néanmoins, leur système de contrôle fait preuve d'une certaine rigidité puisqu'il n'est pas en mesure de prendre en compte les incertitudes. De ce fait, les robots doivent régulièrement réviser leur plan d'actions.

Les constatations que nous venons de réaliser se rapportent à des robots explorateurs évoluant seuls. En effet, bien que plusieurs robots aient déjà été envoyés sur Mars, chacun agit individuellement sans tirer parti de la présence d'autres robots ni développer de comportements coopératifs. Afin d'augmenter les performances des missions futures, les scientifiques prévoient d'envoyer plusieurs robots capables d'agir de manière coopérative. Comme l'expliquent Estlin et al. [Estlin *et al.*, 1999], ceci permettrait tout d'abord d'augmenter la quantité de connaissances récoltées en divisant le travail entre les robots. De plus, il serait possible d'élaborer des robots spécialisés dans des tâches plus particulières et ainsi d'augmenter les capacités de la colonie. Enfin, en cas de défaillance d'un robot (panne ou échec de la réalisation d'une tâche), un autre robot pourrait intervenir afin de réparer la panne ou exécuter la tâche laissée en suspens. L'aptitude des robots à coopérer permettrait alors d'augmenter leurs performances.

La mise en œuvre de colonies de robots pose les mêmes difficultés que celles exposées précédemment pour le développement d'un seul robot (contraintes, caractéristiques de l'environnement, limitations des ressources). Il est de plus nécessaire de gérer les interactions entre les robots et les problèmes de coopération. Ainsi, les contraintes de précédence entre les tâches ajoutent des dépendances entre les robots. La satisfaction de certaines pré-conditions peut, par exemple, dépendre d'autres robots : si un robot $\mathcal{A}g_1$ est spécialisé dans le déploiement de matériel et un second robot $\mathcal{A}g_2$ est chargé des mesures, la réussite du robot $\mathcal{A}g_2$ dépendra du robot $\mathcal{A}g_1$.

Les restrictions de communication concernent également les communications entre robots. Dans le cas des robots sur Mars, ces derniers peuvent communiquer directement entre eux ou via un satellite. Comme nous l'avons précédemment expliqué, la plupart du temps, les robots ne peuvent pas espérer communiquer via le satellite. Toutefois, il leur est possible d'échanger des messages sans passer par le satellite à condition qu'ils soient proches physiquement et que la topologie du site le permette (par exemple qu'aucun obstacle n'empêche la liaison). Lorsque les robots évoluent dans de grands espaces et/ou des terrains accidentés, ce type de communication n'est pas envisageable. En effet, *a priori*, rien ne garantit qu'ils puissent établir une tel échange d'information, il est donc préférable d'éviter le recours à cette forme de communication.

Des difficultés similaires sont répertoriables dans d'autres domaines d'application de la robotique exploratoire. Nous pouvons plus particulièrement citer les domaines ayant trait au développement des colonies de robots démineurs ou de robots sous-marins [Turner, 2005].

4.1.2 Robots rescue

Les robots sauveteurs ou « robot rescue » présentent de grandes similitudes avec les robots explorateurs. L'expansion de ce type d'application a été favorisée par la mise en place de la RoboCup Rescue [Morimoto, 2000, The RoboCup Rescue Technical Committee, 2000], une compétition internationale proposant de développer des robots chargés d'intervenir après une catastrophe naturelle telle qu'un tremblement de terre. Trois types de robots sont engagés dans ces missions : des robots pompiers chargés d'éteindre les feux, des robots ambulanciers devant secourir les victimes, et des robots policiers déblayant les routes. Des dépendances fortes entre les différents types de robots peuvent être identifiées : les policiers doivent avoir déblayé les routes pour que les pompiers puissent accéder aux feux et que les ambulanciers viennent au secours des victimes.

Comme en robotique exploratoire, l'environnement est incertain (incertitude sur les durées d'extinction des feux, de déblayage des routes, ...) et partiellement observable. En effet, chaque robot a une perception limitée à une dizaine de mètres. Il ne connaît donc pas l'état de tous les bâtiments, l'emplacement ou l'état de santé de toutes les victimes. De plus, l'environnement est dynamique (les feux évoluent, la santé des victimes peut se dégrader, ...), séquentiel et continu.

Les communications entre les différents intervenants sont restreintes. Afin de ne pas saturer les liaisons radios, chaque robot ne peut envoyer qu'un nombre limité de messages. Il est également possible de diffuser des messages dans l'environnement mais ces derniers ne sont perçus que dans un périmètre d'une trentaine de mètres. Les ressources des robots sont par ailleurs limitées : un robot pompier ne dispose par exemple que d'une certaine quantité d'eau. Suivant le type de robot la nature des ressources est susceptible de varier, pour les robots pompiers il s'agira d'eau, les ambulanciers quant à eux ne pourront véhiculer qu'un seul blessé à la fois.

Enfin, de nombreuses contraintes temporelles émanent du problème. Tout d'abord, une date de fin de la crise est fixée (300 unités de temps). De plus, des contraintes temporelles intrinsèques sont identifiables : une victime doit être secourue avant qu'elle ne décède, un feu doit être éteint avant que le bâtiment soit complètement brûlé, etc.

4.1.3 Robots dockers

Les robots dockers constituent une application moins répandue des colonies de robots mais très similaire à celles qui viennent d'être exposées. Ces robots, dits de service, sont utilisés pour décharger des bateaux et ranger des containers dans des hangars. Les contraintes temporelles sont

moins importantes que celles évoquées dans les applications précédentes, puisqu'elles consistent à ranger les marchandises avant que le bateau suivant arrive. Néanmoins, des contraintes d'ordre entre les tâches (ou pré-conditions) peuvent intervenir afin d'optimiser le rangement des containers ou répondre à des contraintes d'utilisation. Il est possible d'envisager des situations où les containers de fruits et légumes seront utilisés avant les containers de textile et devront donc être rangés en dernier pour être plus facilement accessibles par la suite. Le temps nécessaire pour décharger et ranger les containers est incertain, il dépend de leur poids et de leur emplacement de stockage. Enfin, la fréquence de rotation des bateaux impose une grande efficacité de la part des robots.

4.1.4 Constellations de satellites

Nous allons terminer cette section par la présentation de systèmes quelque peu différents puisqu'ils sont constitués d'engins autonomes gravitant dans l'espace. Les constellations de satellites sont vouées à l'observation et à la surveillance. Damiani et al. [Damiani, 2005, Damiani *et al.*,] décrivent un projet, inspiré des missions Bird et Fuego, qui utilise un tel système spatial pour la surveillance des feux de forêts et des volcans terrestres. Deux types d'activités peuvent être différenciés : la surveillance automatique pour la détection de nouveaux foyers ou de nouvelles activités volcaniques, et l'observation de zones ciblées à risque ou actives.

L'utilisation de plusieurs satellites permet d'étendre la zone de couverture et d'augmenter la fréquence des observations d'un site donné. A partir des stations au sol, des demandes de suivi sont envoyées aux satellites, leur objectif étant de maximiser le nombre et la pertinence des observations réalisées par la colonie et retournées aux stations.

La communication entre le sol et les satellites n'étant pas possible à tout moment (contraintes physiques, technologiques, financières, etc.), les satellites doivent être en mesure d'agir de manière autonome. Cette autonomie doit également leur permettre de faire face à des situations où le temps de réaction n'est pas suffisant pour faire appel au contrôle au sol. Par ailleurs, elle permet, comme dans les applications précédemment citées, de simplifier le travail réalisé par les opérateurs humains. Mentionnons que les restrictions de communication concernent également les communications de satellite à satellite qui ne sont pas directement possibles et doivent donc passer par un tiers (station au sol ou satellite géostationnaire) accessible par intermittence.

Comme dans le cas des robots sur Mars, la spécificité des composants électroniques utilisés restreint les capacités de calcul des satellites. La mémoire de masse et l'énergie sont également limitées. Chaque observation doit par ailleurs être stockée jusqu'à la prochaine fenêtre de communication avec le sol qui permettra de décharger des données et ainsi libérera la mémoire. En attendant ce déchargement, les satellites doivent donc gérer leur capacité mémoire. La quantité de mémoire consommée par une observation ainsi que son temps de déchargement sont incer-

tains. Puisque les actions sont non déterministes, une part d'incertitude doit donc être prise en compte.

Un grand nombre de contraintes temporelles sont à considérer. Les périodes d'observation de chaque site ainsi que les fenêtres temporelles de déchargement de données peuvent être décrites par des intervalles de temps. Par ailleurs, les données observées ont une durée de validité et doivent donc être communiquées avant d'être périmées. La communication de données pertinentes étant un facteur crucial de la réussite d'une mission, une attention particulière doit être apportée à ces contraintes.

Afin de maximiser la quantité d'informations transmises au sol, les satellites doivent bien entendu coopérer. Une couverture complète et fréquente des sites doit être assurée d'où une nécessaire coordination des orbites des satellites. Chaque centre de mission ne pouvant communiquer qu'avec un seul satellite à la fois, une coordination des communications est également requise.

4.2 Formalisation du problème

La description du contexte applicatif de notre travail nous a permis de mettre en évidence de fortes similitudes entre les applications présentées. Elle contribue ainsi à motiver les caractéristiques des problèmes abordés dans cette thèse et formalisés dans cette section.

4.2.1 Caractéristiques des problèmes considérés

Des systèmes multi-agents coopératifs

Les problèmes auxquels nous nous intéressons [Beynier, 2003, Beynier et Mouaddib, 2004d] peuvent naturellement être modélisés par des **systèmes multi-agents coopératifs**. Ils consistent en effet en un certain nombre d'entités devant réaliser, de manière autonome, un ensemble de tâches dans un même environnement. Le but de ces agents consiste à maximiser une mesure de performance commune : maximiser les connaissances scientifiques utiles récoltées, maximiser un score[25] ou le nombre de containers rangés.

Des contraintes complexes

Différentes contraintes doivent par ailleurs être respectées afin de construire des solutions adaptées aux problèmes posés. Nous avons plus particulièrement pu recenser :

[25] Dans la RoboCup Rescue, un score est calculé en fonction du nombre de victimes sauvées, de la surface de bâtiments endommagés par le feu, ...

- des contraintes temporelles
- des contraintes de précédence
- des contraintes de ressources
- une communication limitée voire impossible
- des capacités de calcul restreintes

Les deux derniers types de contraintes répertoriés influencent plus particulièrement le processus délibératif des agents. Ils imposent que les agents prennent des décisions en limitant les calculs et les communications. Les trois premiers types de contraintes portent sur l'exécution des tâches. Pour chaque tâche, elles définissent des conditions au succès de son exécution.

Les **contraintes temporelles** définissent, pour chaque tâche, une fenêtre temporelle durant laquelle l'exécution de la tâche doit avoir lieu.

Les **contraintes de précédence** traduisent des contraintes d'ordre entre les tâches. Elles permettent la formalisation de pré-conditions telles que « la tâche A doit être terminée pour que la tâche B puisse commencer ». Enfin les **contraintes de ressources** posent les conditions sur les ressources nécessaires à l'exécution d'une tâche.

Remarquons que la prise en compte de ces trois types de contraintes permet de représenter un large spectre des contraintes pouvant régir l'exécution des tâches. Néanmoins, les possibilités de contraindre l'exécution d'une tâche étant multiples, il est tout à fait possible d'identifier d'autres types de contraintes. De ce fait, nous ne considérerons pas cet ensemble comme exhaustif.

Des problèmes de grande taille

Dans la première partie de ce document, nous avons évoqué la nécessité de traiter des problèmes de grande taille. En effet, les approches existantes ne traitent souvent que des problèmes à deux agents et moins d'une dizaine d'actions. Dans les applications que nous venons d'évoquer, il est évident que des problèmes de taille beaucoup plus importante seront à envisager. En ce qui concerne les robots martiens, les opérateurs communiquent une fois par jour l'ensemble des tâches à réaliser pour la journée. Cet ensemble compte en moyenne une centaine de tâches pouvant être exécutées par 3 ou 4 agents. Dans le cas des constellations de satellites, il est possible de répertorier des scénarios composés de 12 satellites devant effectuer plusieurs centaines d'observations. Les problèmes envisagés peuvent donc contenir plus d'une centaine de tâches et une dizaine d'agents.

4.2.2 Mission et environnement

Nous appelons « mission » l'ensemble des agents et des tâches constituant les problèmes qui viennent d'être décrits.

Définition 20 ***Mission***

Une ***mission*** $\mathcal{X}$ *est définie par un couple* $< \mathcal{A}g, \mathcal{T} >$ *tel que :*

- $\mathcal{A}g = \{\mathcal{A}g_1, \cdots, \mathcal{A}g_n\}$ *définit l'****ensemble des*** n ***agents*** $\mathcal{A}g_i \in \mathcal{A}g$.
- $\mathcal{T} = \{t_1, \cdots, t_p\}$ *est l'****ensemble des tâches*** *devant être exécutées par les agents.*

Pour être impliqué dans la mission, chaque agent doit au moins exécuter une tâche. La réalisation d'une mission consiste en l'exécution de l'ensemble des tâches $\mathcal{T}$ par les agents $\mathcal{A}g$. Elle est réalisée au sein d'un environnement $\mathcal{E}$.

Nous supposerons que les tâches sont initialement réparties sur les agents. Nous n'aborderons pas le problème de cette répartition. Mentionnons toutefois, que cette allocation doit tenir compte des aptitudes des agents : si les agents sont spécialisés, l'allocation des tâches doit respecter les compétences de chacun. Elle doit avoir pour résultat un scénario réalisable où chaque agent a la possibilité d'exécuter ses tâches, étant données les contraintes temporelles, les contraintes de précédence et les ressources dont il dispose initialement. Concernant les algorithmes d'allocation des tâches, Hanna et Mouaddib [Hanna et Mouaddib, 2002, Hanna, 2003], et plus récemment Abdallah et Lesser [Abdallah et Lesser, 2005], ont proposé des méthodes basées sur l'utilisation de processus décisionnels de Markov. Nous pouvons également citer les méthodes basées sur la négociation et les contrats proposées par Sandlhom et al. [Sandholm et Lesser, 1995, Sandholm, 1998]. Afin de réaliser l'allocation de tâches pour des robots physiques, des travaux basés sur les principes de ventes aux enchères ont par ailleurs été présentés [Gerkey et Matarić, 2002, Esben *et al.*, 2002].

Présentation de l'environnement

Dans ce travail, nous ne nous préoccupons pas de la manière dont les tâches sont réalisées, nous nous intéressons seulement au fait qu'elles sont exécutées ou non. Nous pouvons donc nous abstraire de l'environnement physique dans lequel évolue l'agent, pour envisager la construction d'un environnement virtuel constitué de l'ensemble des tâches et des agents. Les états de cet environnement sont composés des tâches exécutées et de leurs dates d'exécution. Cet environnement virtuel est :

- **partiellement observable** : chaque agent n'a connaissance que de la réalisation des tâches qu'il a lui même exécutées.
- **stochastique** : il est difficile de modéliser parfaitement l'ensemble des lois régissant l'exécution des tâches, l'incertitude sur les exécutions des tâches permet alors de tenir compte de ces imprécisions. Ainsi, les durées d'exécution et les consommations de ressources des tâches sont considérées comme non déterministes.
- **séquentiel** : en raison, entre autres, des contraintes de précédence, l'exécution d'une tâche influence les actions à venir.

- **statique** : les agents sont les seuls à pouvoir influencer l'état de l'environnement virtuel. Aucun facteur extérieur ne peut le modifier, l'environnement est donc statique.
- **discret** : les durées d'exécution et les consommations de ressources étant discrétisées, la représentation du temps et des ressources est discrète.

4.2.3 Représentation par un graphe acyclique

Une mission peut aisément être représentée par un graphe orienté acyclique où les sommets correspondent aux tâches à réaliser et les arêtes aux contraintes de précédence entre les tâches. Un tel exemple de graphe est décrit par la figure 4.3.

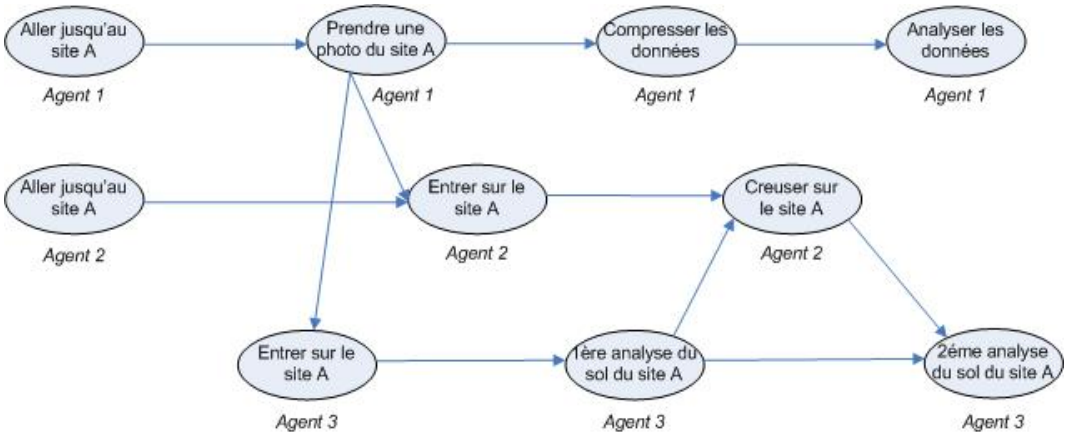


FIG. 4.3 – Exemple d'un graphe de mission

Ce graphe, appelé **graphe de mission**, est nécessairement acyclique puisque dans le cas contraire nous serions en présence d'une boucle traduisant des contraintes telles que « A doit être exécutée avant B, B avant C et C avant A ». Ceci représente bien évidemment un problème insoluble.

Le graphe présenté sur la figure 4.3 décrit un scénario inspiré de la robotique exploratoire sur Mars. Chaque nœud correspond à une tâche. Il met en jeu trois agents : le premier est spécialisé dans la prise de photo, le deuxième est équipé d'une foreuse et le troisième possède des appareils pour l'analyse du sol. Afin que le premier agent prenne en photo le site A, aucun agent ne doit être présent sur le site sinon il risque d'en masquer une partie. Les agents 2 et 3 ne doivent donc pas rentrer sur le site avant que la photo ne soit prise. Remarquons que pour prendre sa photo, l'agent 1 n'a pas besoin d'entrer sur le site A mais seulement de se rendre à proximité de celui-ci. L'agent 3 est chargé de réaliser des analyses du sol en surface et en profondeur. N'étant pas équipé du matériel pour creuser, il doit se coordonner avec l'agent 2 capable de forer le sol.

Par souci de lisibilité, nous ne présenterons que de très petits graphes. Ce formalisme permet toutefois de traiter de plus grands graphes.

Des méthodes de représentation similaires aux graphes de mission ont été utilisées pour la représentation des dépendances entre les tâches dans des systèmes multi-agents. Nous pouvons tout d'abord citer le langage de représentation MAID (Multi-Agent Influence Diagrams) décrit par Koller et Milch [Koller et Milch, 2003] et appliqué dans le cadre des jeux multi-agents. Tout comme nos graphes de mission, les diagrammes d'influence multi-agents représentent les contraintes de précédence par des arêtes reliant les tâches. Nos graphes de mission peuvent être envisagés comme des diagrammes d'influence multi-agents dans lesquels les variables d'utilité et de chance ne seraient pas représentées. Les variables d'utilité de MAID peuvent être assimilées, dans notre formalisme, aux récompenses associées aux tâches. Néanmoins, les diagrammes d'influence multi-agents ne permettent pas de modéliser les contraintes temporelles, ni l'incertitude sur les durées d'exécution.

Les relations de dépendance entre les tâches sont également décrites dans le formalisme TAEMS (Task, Analysis, Environment Modeling and Simulation) présenté par Decker et Lesser [Decker et Lesser, 1993b]. Celui-ci permet la description de la structure des tâches dans les systèmes multi-agents. Il décrit en particulier une relation « enable » correspondant exactement à nos contraintes de précédence. En revanche, les relations appelées « relations de précédence » dans TAEMS n'ont pas la même signification que dans notre travail. Dans le cadre de TAEMS, si une relation de précédence relie t_i à t_j, la tâche t_j ne peut être exécutée que si la tâche t_i est terminée et le résultat de l'exécution de t_i influence la qualité de t_j.

4.2.4 Formalisation d'une tâche

Les choix de représentation qui précèdent nous ont permis de formaliser de manière générale les problèmes que nous souhaitons traiter. Nous allons à présent revenir plus en détail sur les éléments de base du graphe, c'est-à-dire les tâches. Nous allons, dans cette section, nous pencher sur leur formalisation. Nous nous intéresserons tout d'abord à la représentation des différentes contraintes régissant leur exécution, ce qui nous permettra ensuite de définir les caractéristiques d'une tâche.

Représentation des contraintes

L'identification des différents types de contraintes du problème nous a permis de mettre en évidence trois types de contraintes régissant l'exécution des tâches. Nous avons choisi de les formaliser comme des caractéristiques de la tâche elle même.

Les contraintes de précédence définissent, pour chaque tâche t_i, l'ensemble des tâches t_j ($j \neq i$) devant être terminées pour que t_i puisse commencer. Il est alors possible de définir pour

chaque tâche un ensemble $Pred$ de prédécesseurs t_j.

Les contraintes temporelles définissent quant à elles, des fenêtres temporelles sur l'exécution des tâches et peuvent donc être formalisées par des intervalles. Chaque contrainte temporelle est alors représentée par un ensemble de deux valeurs, une date de début au plus tôt et une date de fin au plus tard, définissant les bornes de la fenêtre temporelle. Bien que de nombreuses contraintes temporelles ne soient pas exprimées explicitement par un intervalle de temps précis, elles peuvent aisément être définies comme telles. Ainsi, une tâche devant être réalisée le matin, pourra se voir associer l'intervalle $[6, 12]$. Lorsqu'aucune contrainte temporelle n'est imposée sur une tâche, l'intervalle $[0, \infty]$ lui est associé.

Le troisième type de contraintes concernant l'exécution des tâches porte sur les contraintes de ressources. Un taux initial de ressources est associé à chaque agent au début de la mission. Au fur et à mesure de l'exécution des tâches les ressources diminuent[26]. En effet, chaque tâche consomme une quantité Δ_r de ressources qui doit être inférieure ou égale aux ressources disponibles r[27]. Pour chaque tâche t_i, les contraintes de ressources sont donc définies à partir des consommations de ressources de la tâche et nécessitent que l'égalité $r \geq \Delta_r$ soit respectée.

Définition d'une tâche

Une fois la représentation des contraintes établies, il devient possible de définir les différentes caractéristiques d'une tâche.

Définition 21 ***Tâche***

Chaque ***tâche*** $t_i \in \mathcal{T}$ *est définie par :*

- *un* ***agent*** $\mathcal{A}g_i$ *devant exécuter la tâche.*
- ***différentes durées possibles*** *auxquelles sont associées des* ***probabilités****.* $P_c(\delta_c^i)$ *est la probabilité que l'exécution de la tâche* t_i *dure* δ_c^i *unité de temps.*
- ***différentes consommations de ressources*** *auxquelles sont associées des* ***probabilités****.* $P_r(\Delta_r^i)$ *est la probabilité que l'exécution de la tâche* t_i *consomme* Δ_r^i *unités de ressources.*
- *des* ***contraintes temporelles*** *sur l'exécution de la tâche. On associe à la tâche* t_i*, une fenêtre temporelle* $TC_i = [EST_i, LET_i]$ *restreignant l'exécution de la tâche.* EST_i *désigne la date de début au plus tôt (Earliest Start Time) de la tâche* t_i *et* LET_i *correspond à sa date de fin au plus tard (Latest End Time). L'intervalle d'exécution* I *de la tâche doit donc être inclue dans cette fenêtre temporelle :* $I \subset [EST_i, LET_i]$.

[26] En supposant que les ressources ne soient pas renouvelables.

[27] Pour des raisons de clarté des équations et des notations nous ne considérerons qu'une seule ressource. Cette hypothèse n'affecte ne rien la généricité de notre approche qui eut être étendue afin de tenir compte des consommations de différentes ressources.

- *des **prédécesseurs** : la tâche t_i possède un ensemble de prédécesseurs $Pred_i$ désignant l'ensemble des tâches devant être exécutées afin que t_i puisse commencer.*

$$\forall t_i \in \mathcal{T}, t_i \notin root \quad \Leftrightarrow \exists t_j \in \mathcal{T} : t_j \in Pred(t_i)$$

où root désigne les toutes premières tâches devant être exécutées c'est-à-dire celles n'ayant pas de prédécesseurs.

- *une **récompense** $\mathcal{R}_i$: elle est obtenue par le ou les agents lorsque la tâche t_i est exécutée en respectant les contraintes.*

Les branchements du graphe correspondent par ailleurs à des nœuds « et ». Afin de maximiser la mesure de performance, toutes les tâches d'une mission doivent donc être réalisées.

Nous ne considérons que les problèmes dans lesquels chaque tâche n'est exécutée que par un seul agent $\mathcal{A}g_i$. Si l'exécution d'une tâche nécessite le concours de plusieurs agents (par exemple déplacer un cube), alors elle peut être dupliquée autant de fois qu'il y a d'agents y participant. Il est cependant nécessaire d'ajouter des contraintes imposant la simultanéité de l'exécution de ces duplicata de tâches.

Les consommations de ressources et les durées d'exécution étant incertaines, chaque tâche possède différentes durées et consommations de ressources possibles auxquelles sont associées des probabilités.

La définition des probabilités sur la durée d'une tâche et sa consommation de ressources supposent qu'elles sont indépendantes. Néanmoins, cette hypothèse n'affecte en rien la généricité de notre approche qui peut aisément être adaptée au cas où ces quantités sont dépendantes l'une de l'autre. On considérera alors une distribution de probabilités sur les couples (Δ_r^i, δ_c^i) telle que $P_{rc}(\Delta_r^i, \delta_c^i)$ est la probabilité que l'exécution de la tâche prenne δ_c^i unités de temps et consomme Δ_r^i ressources. Un tel exemple de dépendance entre les consommations de ressources et les durées d'exécution est décrit dans [Beynier *et al.*, 2006].

Lorsqu'une tâche est exécutée avec succès, en respectant les contraintes imposées, les agents obtiennent une récompense $\mathcal{R}_i$. Cette récompense ne dépend ni du temps, ni des ressources consommés. Il est cependant tout à fait possible de considérer une fonction de récompense fondée sur de tels paramètres. Étant donnée une mission $\mathcal{X}$, l'objectif des agents consiste à maximiser la somme des récompenses obtenues. Remarquons que dans le cas où la récompense d'une tâche est octroyée à l'agent l'ayant exécutée et non à la colonie, le critère de performance se traduit par la maximisation de la somme des récompenses obtenues par chaque agent.

La figure 4.4 décrit un graphe de mission et associe à chaque nœud les caractéristiques de la tâche qu'il représente. Nous considérons des graphes de mission dans lesquels les tâches de chaque

agent sont complètement ordonnées. Bien que certaines missions ne présentent pas au premier abord cette propriété, un ordre total peut généralement être déduit à partir des contraintes de précédence et des contraintes temporelles. En effet, les relations de précédence établissent un ordre partiel sur les tâches d'un même agent. Celui-ci peut dans bien des cas être étendu à un ordre total en prenant en compte les contraintes temporelles. Considérons deux tâches t_i et t_j non ordonnées et contraintes respectivement par les fenêtres d'exécution $[6, 12]$ et $[4, 7]$, nous pouvons naturellement imposer que t_j précède t_i.

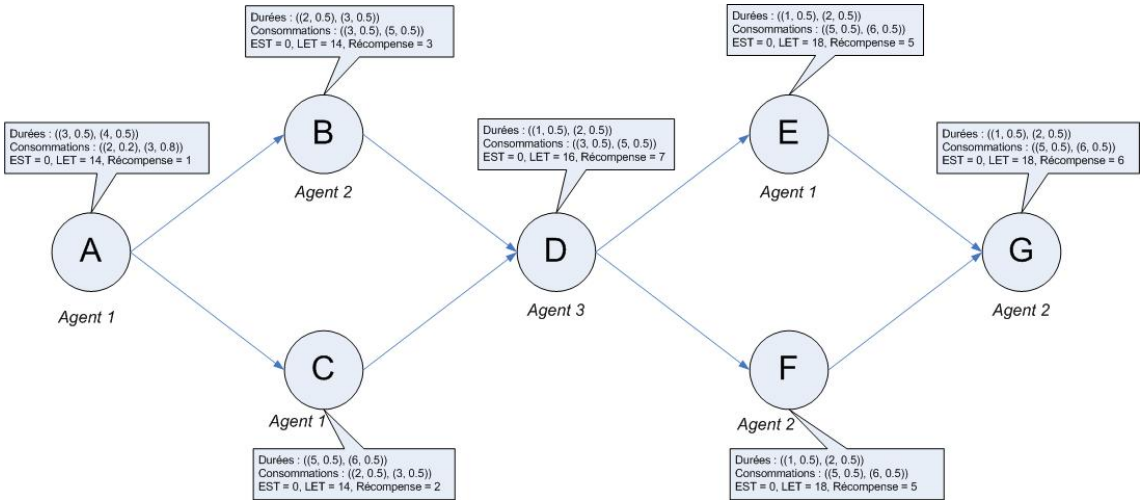


FIG. 4.4 – Exemple détaillé d'une mission

4.3 Résolution par planification

Après avoir formalisé les tâches composant nos problèmes, nous allons achever ce chapitre en nous intéressant à la deuxième et dernière composante des missions considérées : les agents. Étant donnée une mission $\mathcal{X}$, il est nécessaire de mettre en œuvre un processus de décision (ou processus de délibération) permettant aux agents de décider quelle action exécuter et à quel moment.

4.3.1 Le cahier des charges

Les applications présentées au début de ce chapitre ont permis d'identifer d'importantes difficultés concernant l'échange de messages durant l'exécution des tâches. La communication étant difficile et non garantie, il semble primordial de développer un processus décisionnel ne nécessitant **pas de communication directe**.

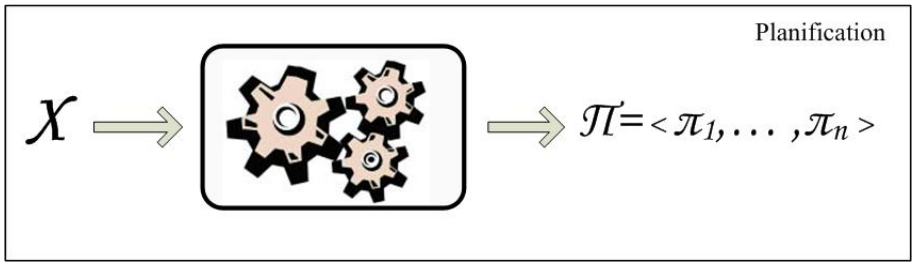


FIG. 4.5 – Schématisation du processus de planification

Les agents étant de plus répartis dans l'espace durant l'exécution de la mission, il n'est pas possible de faire appel à une entité centrale chargée de prendre et coordonner les décisions. Les agents doivent donc décider comment agir de manière **autonome** et **décentralisée**.

Enfin, étant donnée la limitation des ressources computationnelles des agents, ce processus décisionnel doit faire preuve d'une **faible complexité algorithmique**. La prise de décision doit donc être peu « gourmande » en temps et en espace mémoire.

4.3.2 L'approche adoptée

Rappelons que nous avons adopté le point de vue de la rationalité afin de définir l'intelligence d'un système. Étant donnée l'incertitude émanant des problèmes considérés et la forme de la mesure de performance (une maximisation de récompenses), l'approche que nous avons adoptée repose sur le principe de **maximisation de l'utilité espérée**.

Nous avons ainsi choisi de développer une solution basée sur un **processus de planification** des actions des agents permettant une maximisation de la mesure de performance traduite par l'utilité espérée.

Afin de limiter les calculs réalisés par les agents, et de leur offrir une grande réactivité, nous avons opté pour **un processus de planification hors-ligne**, c'est-à-dire antérieur à la phase d'exécution des actions. Le cycle de vie des agents peut alors être divisé en deux phases.

La première appelée phase « hors-ligne » permet à l'agent de planifier son comportement et de déterminer une stratégie d'action de sorte de maximiser l'utilité espérée du système. Nous nous plaçons dans le cadre général des systèmes multi-agents pour lesquels aucun contrôle centralisé n'est possible. Il est alors nécessaire que chaque agent puisse prendre ses décisions de manière autonome. A partir d'une mission $\mathcal{X}$, la planification doit permettre de calculer une politique pour chaque agent. La figure 4.5 schématise ce processus dont la mise en oeuvre constitue le cœur de notre travail. Précisons que le processus de planification qui sera développé doit être en mesure de gérer les incertitudes ainsi que les contraintes précédemment recensées.

La seconde phase dite « en-ligne » consiste en l'exécution des actions par les agents, à partir des politiques précédemment calculées. Les agents appliquent alors leurs propres stratégies à partir des observations partielles du système et sans communiquer. Le cycle de vie de l'agent durant cette phase peut être décrit comme une boucle perception-action identique à celle présentée en première partie. L'agent perçoit son environnement et déduit à partir de sa politique quelle action exécuter. La complexité en temps de cette délibération est donc réduite. En ce qui concerne la quantité de mémoire requise, cette dernière est dépendante de la taille de la politique de l'agent.

La figure 4.6 représente l'articulation des deux phases constituant le cycle de vie d'un agent. Une fois la phase d'exécution des tâches terminée, une nouvelle phase de planification peut commencer afin de calculer la stratégie d'exécution de nouvelles tâches.

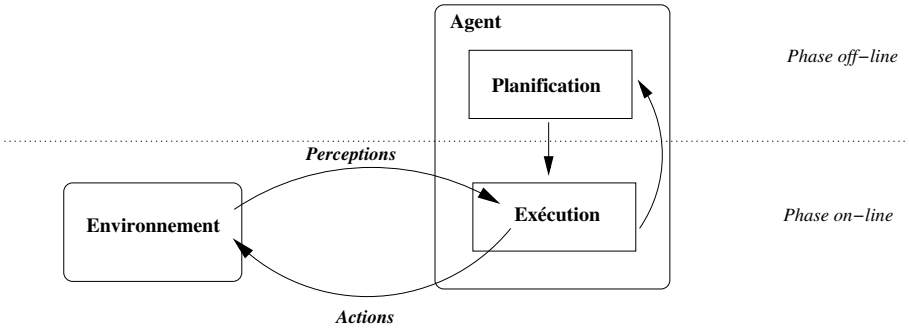


FIG. 4.6 – Phases de planification et d'exécution

Nous tenons à préciser que l'impossibilité de communiquer pendant l'exécution des tâches ne signifie pas que les agents ne puissent pas communiquer avant l'exécution afin d'établir un plan d'actions. Dans le cas des robots martiens, une fois que les communications avec la Terre sont terminées, les robots peuvent profiter du fait que le satellite soit encore correctement positionné, pour échanger des messages hors-ligne.

4.3.3 La planification stochastique

Une synthèse de différentes méthodes de planification stochastique a été réalisée par Blythe [Blythe, 1999b, Blythe, 1999a]. Celui-ci distingue les approches markoviennes et les méthodes basées sur les techniques de planification classiques telles que STRIPS ou GPS. Cette deuxième classe d'approches consiste en l'adaptation des méthodes classiques afin de gérer différents résultats possibles pour l'exécution d'une même action. Elles reposent le plus souvent sur l'utilisation d'opérateurs logiques dont l'application successive décrit un plan permettant ou non d'atteindre un état but. Ces approches cherchent de manière générale un plan d'actions assurant une certaine

probabilité de succès. Différentes méthodes permettent l'obtention d'un plan décrivant une suite d'actions à réaliser. Lors de l'exécution du plan, si l'agent se trouve dans une situation où la prochaine action ne peut être appliquée, le plan devient obsolète. Ces déviations dues aux incertitudes de l'environnement, nécessitent que l'agent procède à une re-planification des actions qu'il lui reste à exécuter. Afin de limiter la re-planification, des plans contingents peuvent être calculés. Un plan contingent peut être représenté par un arbre dans lequel les branches représentent les évolutions les plus probables du système[28]. Ainsi, un comportement plus robuste est obtenu, les agents pouvant faire face à un certain nombre de déviations. Bresina et Washington [Bresina et Washington, 2000] présentent notamment une approche reposant sur le calcul de l'utilité des plans suffixes et permettant la gestion de contraintes temporelles. Néanmoins, toutes ces approches ne garantissent pas un comportement optimal et rationnel des agents et nécessite des phases de re-planification, donc des calculs, lors de l'exécution.

Les processus de décision markoviens offrent une seconde alternative à la planification sous incertitude. Comme nous l'avons décrit en première partie, ces modèles expriment l'objectif de l'agent en terme de maximisation d'une fonction d'utilité espérée. De ce fait, ils permettent l'obtention d'un comportement optimal et rationnel de la part de l'agent. De plus, la politique obtenue fait correspondre une action à tout état possible du système. Ainsi, aucune re-planification n'est nécessaire au cours de la phase d'exécution des tâches. Les processus de décision markoviens présentent donc un cadre de travail tout à fait adapté à la résolution des problèmes qui nous sont posés. Travaillant sur des systèmes multi-agents dans lesquels le contrôle est décentralisé, nous aurons par conséquent recours à l'utilisation des processus décisionnels de Markov Décentralisés.

Bresina et al. [Bresina *et al.*, 2002] soulignent cependant quelques difficultés vis à vis de la formalisation des problèmes de robotique exploratoire par des modèles markoviens. Ils insistent en particulier sur le fait que les travaux existant portant sur les processus décisionnels markoviens considèrent les actions comme instantanées et ne permettent pas la représentation de contraintes temporelles explicites sur l'exécution. Afin que nous puissions planifier l'exécution des missions qui ont été présentées dans ce chapitre, il nous sera donc nécessaire d'étoffer les modélisations markoviennes existantes et de mettre en place une modélisation adaptée du temps et des actions permettant la gestion des contraintes qui ont été recensées. Cette question fera l'objet du prochain chapitre.

[28] Un plan contingent représentant toutes les situations possibles est équivalent aux politiques traités dans les processus décisionnels de Markov.

Conclusion

Dans ce chapitre, nous avons commencé par décrire différents domaines d'application des colonies de robots autonomes, sujet à l'origine de cette thèse. Nous avons ainsi dégagé les caractéristiques de ces systèmes et nous avons motivé la formalisation des problèmes auxquels nous nous intéressons. Nous avons également établi le cadre de notre travail : la planification dans des systèmes multi-agents coopératifs agissant en environnement complexe.

Afin de permettre la planification hors-ligne de l'exécution d'une mission, la modélisation du problème sous forme de DEC-POMDPs a été retenue. Ces modèles ne proposent cependant pas une modélisation adaptée du temps et des actions. Ces lacunes sont à mettre en relation avec les difficultés identifiées au chapitre 3, lors de l'étude des approches pour la résolution de DEC-POMDPs. Nous avions alors montré que ces formalismes considéraient le plus souvent des actions instantanées et ne géraient pas de contraintes temporelles.

Afin de résoudre les problèmes de planification qui nous sont posés, nous allons donc tout d'abord devoir nous pencher sur la mise en place d'une modélisation appropriée permettant en particulier la gestion du temps, de l'incertitude, de l'observabilité partielle et des contraintes du problème.

Chapitre 5

OC-DEC-MDP : une classe de DEC-MDPs contraints

Dans le chapitre précédent, nous avons ébauché une modélisation des problèmes auxquels nous nous intéressons. Nous avons, entre autres, introduit une représentation des tâches et des contraintes. Nous avons ensuite présenté le cœur de notre travail : le contrôle de l'exécution des missions. Nous allons poursuivre cette modélisation en nous plaçant dans l'optique de la planification pour des agents distribués à l'aide de DEC-MDPs[29].

Comme nous l'avons mentionné précédemment, l'emploi des DEC-MDPs pour la planification d'une mission nécessite l'enrichissement de la modélisation du temps et des actions. Nous allons aborder ce chapitre en développant cette problématique, ce qui nous amènera à définir une nouvelle classe de DEC-MDPs, les OC-DEC-MDPs.

Nous nous intéresserons ensuite à l'influence des contraintes sur les composantes des OC-DEC-MDPs. Nous proposerons alors une formalisation des états, des actions, des transitions et des récompenses du problème, permettant leur prise en compte. Après avoir détaillé ces composantes, nous serons en mesure d'extraire les propriétés des problèmes traités et d'en déduire leur complexité.

Nous achèverons ce chapitre par la description d'une méthode pour la construction automatique de ce modèle à partir d'une quelconque mission.

5.1 Modélisation par OC-DEC-MDP

Les durées d'exécution des tâches envisagées étant incertaines, nous avons précédemment évoqué la nécessité d'enrichir la modélisation du temps et des actions.

[29] Nous motiverons par la suite le recours aux DEC-MDPs plutôt qu'aux DEC-POMDPs.

5.1.1 Modélisation du temps et des actions

Les approches s'intéressant aux DEC-MDPs représentent des problèmes dans lesquels toutes les actions ont une même durée de 1 unité de temps. Ainsi, lorsque les agents entreprennent, à l'instant t, l'exécution d'une action jointe $\mathcal{A} =< \mathcal{A}_1, \cdots, \mathcal{A}_n >$, toutes les actions $\mathcal{A}_i$ se terminent à l'instant $t+1$. Chaque instant t correspond donc, pour chaque agent, à une étape de décision.

Lorsque les actions des agents ont différentes durées possibles, il n'est pas réalisable de définir une succession d'étapes de décision simultanées à tous les agents. Initialement, tous les agents prennent leur première décision en même temps à t_0, mais les exécutions des actions résultant de ces décisions ne se terminent pas au même instant. A $t_0 + 1$, certains agents auront terminé l'exécution de leur action et d'autres seront encore en train de les réaliser. Il n'est donc pas possible de définir une action jointe comme un ensemble d'actions commençant à t et s'achevant à $t+1$.

Afin de modéliser des problèmes où les durées d'actions sont différentes, il est possible de définir les actions jointes $\mathcal{A} =< \mathcal{A}_1, \cdots, \mathcal{A}_n >$ non pas comme l'ensemble des actions que les agents vont exécuter mais comme l'ensemble des actions que les agents vont exécuter ou sont en train d'exécuter. Considérons par exemple trois agents $\mathcal{A}g_i$, $\mathcal{A}g_j$ et $\mathcal{A}g_k$. A l'instant t, l'agent $\mathcal{A}g_i$ vient de terminer l'action a_i et décide d'exécuter a_{i+1}. Supposons en revanche que les agents $\mathcal{A}g_j$ et $\mathcal{A}g_k$ soient toujours en train de réaliser les actions a_j et a_k commencées antérieurement à t. L'action jointe a est alors décrite par l'ensemble $< a_{i+1}, a_j, a_k >$. Cette méthode permet de définir une action jointe pour chaque instant t sans pour autant supposer que chaque agent exécute une nouvelle action.

Néanmoins, le résultat de l'exécution d'une telle action jointe dépend de l'état d'avancement des actions la composant. En effet, le résultat de l'exécution de l'action jointe a dépend du temps déjà passé à exécuter a_j et a_k. Si $a =< a_{i+1}, a_j, a_k >$ le problème n'est pas markovien puisqu'il est nécessaire de connaître les actions jointes précédentes pour savoir depuis combien de temps chaque action individuelle est exécutée et déduire les probabilités de transition.

Afin de représenter les problèmes sous forme markovienne, il est possible de définir les actions jointes comme un ensemble de couples (a_i, Δ_c^i) où a_i correspond à l'action d'un agent et Δ_c^i décrit la durée d'exécution déjà écoulée de a_i. Supposons dans notre exemple précédent qu'à l'instant t, l'exécution de a_j ait commencé depuis 2 unités de temps, et celle de a_k depuis 3 unités de temps. L'action jointe est alors définie comme $a =< (a_{i+1}, 0), (a_j, 2), (a_k, 3) >$. Ce principe a été utilisé par Mouaddib et Zilberstein [Mouaddib et Zilberstein, 1998] dans le cadre de la planification de PRU[30] à l'aide d'un MDP.

[30] Progressive Processing Unit

Un tel formalisme permet de déterminer l'évolution du système à partir de l'état courant et de l'action jointe exécutée, d'où une modélisation markovienne. Cependant, dans le cadre multi-agent, cette représentation conduit à une explosion de l'espace des actions. En effet, il est nécessaire de considérer tous les couples (a_i, Δ_c^i) possibles et toutes les combinaisons de ces couples.

La prise en compte de l'état d'avancement des actions augmente la taille de l'espace d'actions et par conséquent l'espace de recherche des politiques. Il n'est donc pas envisageable de résoudre de tels problèmes à part pour de très petits nombres de tâches et d'agents. En effet, la résolution des DEC-POMDPs et DEC-MDPs est déjà un problème difficile lorsque les actions jointes ne sont composées que des actions a_i ; l'augmentation du nombre de politiques ne peut que complexifier la recherche. Une telle modélisation du temps et des actions conduit donc à des problèmes très difficiles voire impossibles à résoudre. Remarquons que l'état d'avancement des actions peut également être considéré comme une caractéristique de l'état des agents. Cette représentation conduit alors à une explosion de l'espace des états du système.

5.1.2 Décomposition en MDPs locaux

Afin de développer une modélisation permettant de remédier aux problèmes liés à la représentation du temps et des actions, nous avons choisi de décomposer le problème. Nous proposons alors de le modéliser à l'aide d'un ensemble de MDPs locaux [Beynier et Mouaddib, 2005b]. Cette approche est illustrée par la figure 5.1 dans le cas d'un système à 3 agents. Nous avons nommé ce modèle OC-DEC-MDP du nom de la technique de résolution exposée ultérieurement.

Comme nous l'avons décrit au chapitre précédent, les agents évoluent dans un environnement virtuel. De plus, chaque agent sait quelles tâches il a réalisées et quelles sont celles qu'il lui reste à réaliser. L'état de l'agent est alors localement totalement observable. De ce fait, la décomposition est réalisée sous forme d'un ensemble de MDPs et non pas de POMDPs. C'est pour cette même raison que nous limitons nos propos aux DEC-MDPs.

Chaque MDP modélise le problème décisionnel d'un agent. Ainsi nous n'envisageons pas le problème au niveau collectif mais au niveau individuel. Nous ne traitons alors pas les actions jointes mais les actions individuelles des agents ce qui évite les problèmes liés à la taille de l'espace d'actions. De même, l'espace d'états, la fonction de transition et la fonction de récompense sont décomposées de façon à ne tenir compte que des actions individuelles de l'agent.

Définition 22 *Un **OC-DEC-MDP** pour n agents est constitué d'un ensemble de n MDPs locaux, un pour chaque agent. Le MDP local d'un agent $\mathcal{A}g_i$ est défini par un tuple $< \mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{R}_i >$ tel que :*

- *$\mathcal{S}_i$ est l'ensemble fini des états de l'agent $\mathcal{A}g_i$.*
- *$\mathcal{A}_i$ est l'ensemble fini des actions de l'agent $\mathcal{A}g_i$.*

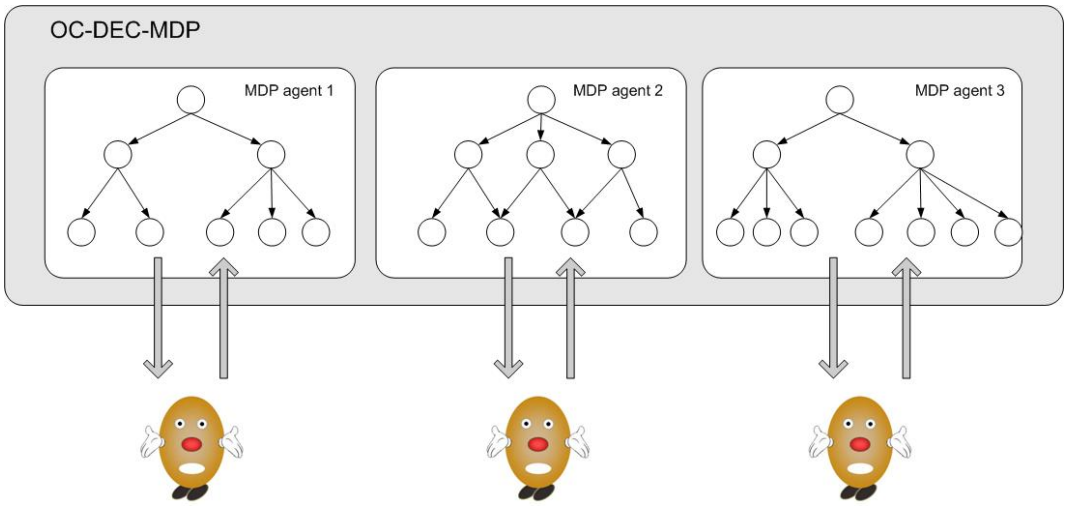


FIG. 5.1 – Illustration de la décomposition en MDPs locaux

- $\mathcal{P}_i$ *définit la fonction de transition de l'agent* $\mathcal{A}g_i$.
- $\mathcal{R}_i : \mathcal{T}_i \rightarrow \mathbb{R}$ *décrit la fonction de récompense de l'agent.* $\mathcal{R}_i(t_i)$ *est la récompense obtenue par l'agent lorsque la tâche* t_i *a été exécutée en respectant les contraintes.*

Les graphes de mission considérés étant finis, nous considérerons des MDPs locaux à horizons finis.

La décomposition du problème permet d'envisager une résolution locale de chaque MDP. Au lieu de résoudre un DEC-MDP de taille conséquente, le problème est alors considéré comme consistant en la résolution d'un ensemble de MDPs. A partir du MDP local d'un agent $\mathcal{A}g_i$, une politique d'actions de l'agent est déduite. Lors de l'exécution des tâches, l'agent n'aura besoin que de cette politique afin de prendre ses décisions.

Néanmoins, les contraintes temporelles et de précédence entre les tâches conduisent à des dépendances entre les agents et par conséquent entre les MDP locaux. Afin que les agents adoptent un comportement coopératif et coordonné, il est donc nécessaire que les MDPs ne soient pas résolus de manière indépendante. Le calcul d'une politique pour chaque agent constituera le sujet principal de notre prochaine partie. Nous proposerons alors une méthode pour la coordination des MDPs locaux. Dans ce chapitre nous nous limiterons à la définition du modèle.

5.2 Introduction de contraintes dans les DEC-MDPs

Comme nous l'avons suggéré précédemment, les travaux portant sur les DEC-MDPs se sont très peu intéressés à la gestion de contraintes. Dans cette section, nous montrons comment les différents types de contraintes recensées au chapitre précédent influencent les composantes d'une modélisation sous forme de DEC-MDP. Nous limiterons nos propos à l'ensemble $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ bien qu'ils puissent être étendus à l'ensemble des observations et à la fonction d'observation.

5.2.1 Contraintes temporelles

Les contraintes temporelles ont pour effet de restreindre les intervalles d'exécution possibles de chaque tâche. En effet, elles définissent des bornes sur leurs dates de début et de fin d'exécution. Les situations d'exécution ne respectant pas ces contraintes n'ont donc pas à être envisagées d'où, une restriction des espaces d'états et d'actions de chaque agent.

Les contraintes temporelles influencent également le passage d'un état à un autre, elles doivent donc être prises en compte lors de la définition de la fonction de transition. Lors de l'exécution d'une tâche, une récompense est en effet attribuée au système à condition que les contraintes temporelles sur la tâche aient été respectées. Dans le cas contraire, la tâche est considérée comme ayant échoué et aucune récompense n'est obtenue.

5.2.2 Contraintes de précédence

Les contraintes de précédence imposent un ordre partiel sur l'exécution des tâches. Ce dernier, combiné aux contraintes temporelles et aux durées d'exécution, permet de restreindre d'autant plus les dates de début possibles de chaque tâche et leurs intervalles d'exécution. En effet, toute tâche t_i ne peut être effectuée avant ses prédécesseurs, ce qui limite fortement ses possibilités d'exécution. Les espaces d'états et d'actions se trouvent par conséquent directement influencés par les contraintes de précédence.

Par ailleurs, le passage d'un état à un autre dépend du respect de ces contraintes : si les prédécesseurs n'ont pas terminé, l'exécution de la tâche échouera et aucune récompense ne sera obtenue. Les contraintes de précédence interviennent donc aussi dans la définition de la fonction de transition et l'attribution des récompenses.

5.2.3 Contraintes de ressources

La disponibilité de ressources suffisantes est une condition nécessaire à l'exécution d'une tâche. Il n'est donc pas possible d'envisager des situations où un agent exécuterait une tâche en n'ayant plus ou pas assez de ressources.

De plus, le manque de ressources lors de l'exécution d'une tâche conduit avec certitude à un échec dans lequel aucune récompense n'est obtenue. Par conséquent, tous les éléments du tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ sont influencés par les contraintes de ressources.

Les différentes contraintes considérées influencent donc sans exception les composantes d'une modélisation sous forme de DEC-MDP. Dans le cadre des OC-DEC-MDPs, chaque élément des tuples $< \mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{R}_i >$ composant les MDPs locaux, doit donc être défini de façon à rendre compte de ces contraintes.

5.3 Composantes du modèle

La décomposition du problème en MDPs locaux passe par la définition, pour chaque agent, d'un espace d'états, d'un espace d'actions, d'une fonction de transition et d'une fonction de récompense. Nous allons à présent revenir plus en détail sur chacune de ces composantes. Comme l'a montré la section précédente, chacun de ces éléments devra être soigneusement défini afin de tenir compte des contraintes sur l'exécution des tâches.

5.3.1 Espace d'états

L'espace d'états de chaque agent (ou espace d'états du MDP local) peut être décomposé en trois parties distinctes faisant référence aux trois types d'états suivants :

- les états de succès comprenant l'état initial
- les états d'échec partiel
- les états d'échec total

États de succès

Considérons un agent $\mathcal{A}g_i$ venant de terminer avec succès l'exécution d'une tâche et devant décider de sa prochaine action. L'agent se trouve dans ce que nous appelons un « état de succès » : il vient de réaliser une tâche t_i pendant un intervalle I. Il lui reste également une quantité r de ressources. La décision de l'agent concernant l'exécution de sa prochaine tâche t_{i+1} dépend de ces trois paramètres. Afin que chaque agent ait une observabilité totale de son état, la description de l'état doit donc en tenir compte.

Un état succès est alors représenté par le triplet $[t_i, I, r]$, où t_i est la dernière tâche exécutée, r est la quantité de ressources encore disponible, et I l'intervalle d'exécution de t_i. Bien que la date de début de l'intervalle I ne soit pas directement exploitée par la suite, elle fournit une information supplémentaire pouvant s'avérer précieuse en vue des extensions de ce travail. Elle

nous informe en particulier sur la date de fin des prédécesseurs de t_i et est, de ce fait, intégrée à la description des états.

Initialement, l'agent se trouve dans un état de succès. Il n'a réalisé aucune tâche, nous désignons alors son état initial par le triplet $[initial, R_{ini}, [start\_time, start\_time]]$ où R_{ini} est la quantité de ressources initiale de l'agent, et $start\_time$ la date à laquelle la mission commence. Cet état peut être considéré comme résultant de l'exécution de la tâche fictive nommée « initial ». Cette exécution se termine à $start\_time$, la première tâche réelle de l'agent peut alors commencer.

États d'échec partiels

Lorsque l'agent $\mathcal{A}g_i$ commence l'exécution d'une tâche t_{i+1}, il est possible qu'il échoue car les prédécesseurs de celle-ci n'ont pas terminé. L'agent arrive alors dans un état échec dont il peut sortir en ré-essayant ultérieurement d'exécuter la tâche. Un tel échec est donc appelé échec partiel.

Nous supposons que le fait d'essayer d'exécuter la tâche et de se rendre compte que les prédécesseurs n'ont pas terminé demande une unité de temps. Lorsque l'agent tente d'exécuter la tâche à st, il réalise donc qu'il a échoué partiellement à $st + 1$.

Un échec partiel consomme des ressources. Si l'agent dispose à st de r' ressources, après un échec partiel il lui restera une quantité r de ressources telle que $r = r' - \Delta'_r$ où Δ'_r désigne la consommation de ressources de l'échec partiel.

Un état d'échec partiel est défini par un tuple $[t_i, I, et(I'), r]$ où t_i correspond à la dernière tâche menée à son terme par l'agent et $et(I')$ désigne la date de fin de t_i. $I = [st, st + 1]$ décrit la fenêtre temporelle durant laquelle a eu lieu l'échec et r est la quantité de ressources disponibles après cet échec. Il est important de mémoriser la dernière tâche exécutée ainsi que sa date de fin car ces données peuvent influencer la décision. Nous reviendrons par la suite plus en détail sur ce point.

La figure 5.2 présente les états d'échecs partiels et de succès associés à chaque tâche de la mission précédemment décrite par la figure 4.4.

États d'échec total

Lorsque l'agent manque de ressources ou ne respecte pas les contraintes temporelles sur l'exécution de la tâche, il passe dans un état d'échec total.

A chaque tâche correspond un unique état d'échec total dans lequel l'agent aboutit quelle que soit la date ou la cause de l'échec. L'espace d'état d'un agent compte donc autant d'états d'échec total que de tâches devant être réalisées par l'agent. Les états d'échec total sont considérés comme terminaux.

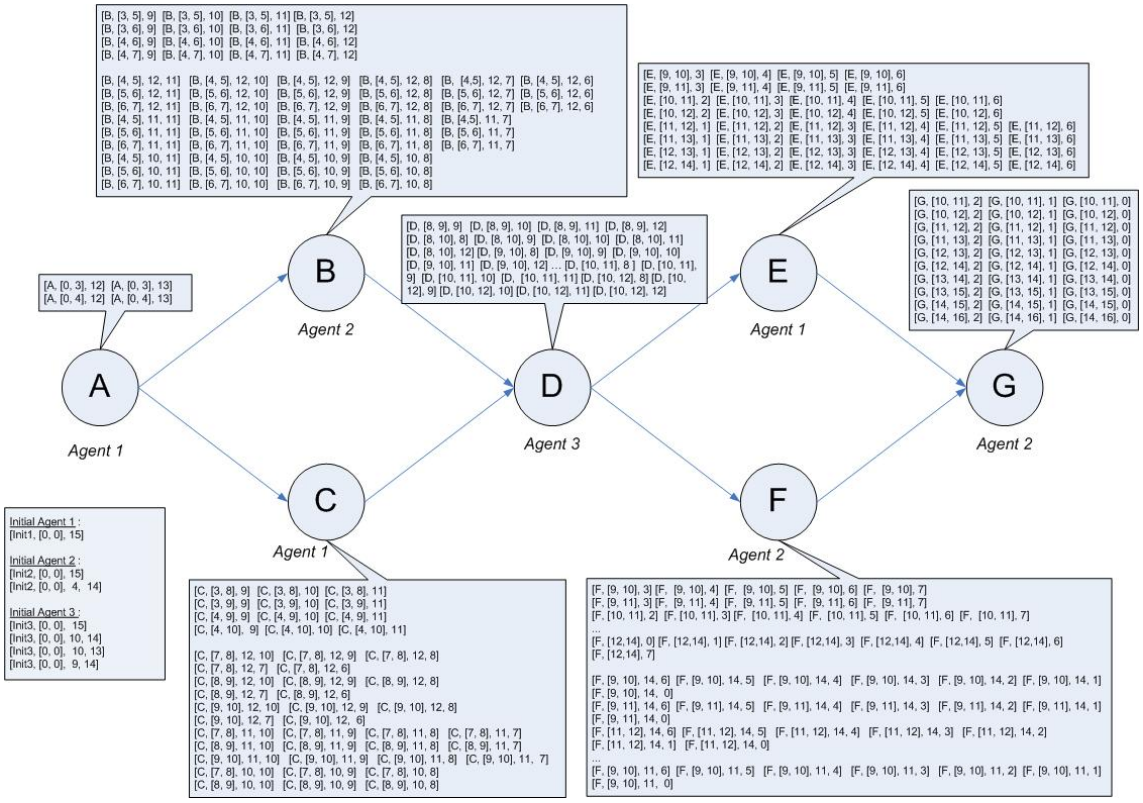


FIG. 5.2 – États associés à chaque tâche d'une mission

Étant donné la définition des états, la construction de l'espace d'états doit passer par le calcul des intervalles d'exécution et des taux de ressources pouvant être associés à chaque tâche. Les contraintes temporelles entraînent des restrictions sur ces intervalles. Les taux de ressources disponibles après l'exécution de chaque tâche doivent quant à eux, respecter les contraintes de ressources. Afin de construire un espace d'états tenant compte des contraintes, celles-ci doivent être considérées lors du calcul des intervalles et des taux de ressources possibles pour chaque tâche.

5.3.2 Espace d'actions

Toute action consiste en l'exécution d'une tâche t_i à un instant st, noté $E(t_i, st)$. Nous avons précédemment montré que les dates de début possibles de chaque tâche sont restreintes par les contraintes temporelles et de précédence. Il en a donc de même pour les actions.

Les durées d'exécution des tâches et leurs consommations de ressources étant incertaines, les actions sont non déterministes.

Nous considérons des problèmes dans lesquels l'ensemble des tâches est fini et où les durées d'exécution des tâches et les consommations de ressources sont discrétisées en un nombre fini de valeurs. Les tâches « racine » du graphe ne possédant par ailleurs pas de prédécesseurs, elles peuvent commencer dès que le système « se réveille » à condition de respecter les contraintes temporelles. Soit $start\_time$ cette date de réveil, t_i une tâche racine du graphe et EST la date de début au plus tôt de t_i. Comme nous le verrons en section 5.5, le maximum entre $start\_time$ et EST constitue l'unique date de début de t_i. La tâche t_i ne peut en effet commencer avant au risque de violer les contraintes temporelles. De plus, retarder l'exécution de t_i à une date ultérieure n'aura pour seul effet que de faire perdre du temps aux agents et donc diminuer leur espérance de gain. Il n'y a donc aucun intérêt à commencer plus tard. Nous pouvons conclure de ce raisonnement qu'il existe une unique date de début pour chaque tâche racine du graphe. Par propagation des contraintes temporelles (cf. section 5.5), nous obtenons un nombre fini de dates de début pour chaque tâche. Nous pouvons en déduire que l'ensemble des actions est fini.

5.3.3 Fonction de transition

De la même façon que nous avons identifié trois catégories d'états, trois types de transitions ont été répertoriés : les transitions dues à une exécution réussie, les transitions résultant d'un échec partiel et les transitions menant à un échec total. Nous reviendrons plus en détail, dans la partie suivante, sur le calcul des probabilités de transition. Nous allons pour le moment nous limiter aux facteurs influençant les transitions entre états.

Exécution réussie

Pour qu'un agent $\mathcal{A}g_i$ réussisse l'exécution de sa tâche t_{i+1}, l'exécution des prédécesseurs de t_{i+1} doit être terminée, ce qui suppose que :

- les prédécesseurs aient eu assez de ressources,
- et qu'ils aient terminé leur exécution à st en respectant les contraintes temporelles.

Il est de plus nécessaire que l'agent ait assez de ressources pour exécuter t_{i+1} et que l'exécution de la tâche respecte les contraintes temporelles. Pour ce faire, la date de début de son exécution doit être inférieure ou égale à la date de début au plus tard LST_{i+1} (Latest Start Time) de t_{i+1}. La date de fin de l'exécution doit de plus être inférieure ou égale à LET_{i+1} (Latest End Time).

Échec partiel

Lorsqu'un agent entreprend l'exécution d'une tâche t_{i+1} à st[31] la probabilité qu'il échoue partiellement est égale à la probabilité que l'exécution de ses prédécesseurs ne soit pas terminée

[31] Afin de ne pas surcharger les formules, nous omettrons de préciser le nom de la tâche à laquelle se rapporte st, lorsque celui-ci est implicite. Dans le cas présent, nous utiliserons la notation st au lieu de $st(t_{i+1})$.

et que l'agent ait assez de ressources pour se rendre compte de son échec partiel. Si l'agent manque de ressources lors de l'échec partiel, il passe dans un état d'échec total.

Les prédécesseurs peuvent ne pas avoir terminé leur exécution car ils termineront plus tard ou bien ils ne termineront jamais. Dans ce dernier cas, au moins un des prédécesseurs a échoué totalement : il a manqué de ressources ou a violé les contraintes temporelles.

Remarquons que si st correspond à la dernière date de début possible de la tâche ($st = LST_{i+1}$) et que les prédécesseurs n'ont pas terminé, l'échec partiel se transforme en échec total. En effet, l'agent ne pourra pas retenter l'exécution de la tâche ultérieurement. Pour que l'agent passe dans un état d'échec partiel, il est nécessaire que st soit strictement inférieur à LST_{i+1}.

Échec total

• **Manque de ressources :** La probabilité qu'un agent échoue par manque de ressources et qu'il arrive pour cette raison dans un état d'échec total dépend des probabilités sur les consommations de ressources.

Si les prédécesseurs n'ont pas terminé leur exécution, la probabilité que l'agent manque de ressources est égale à la probabilité que l'échec partiel consomme plus de ressources que celles disponibles.

Si les contraintes de précédence et temporelles sont respectées, la probabilité de manquer de ressources est égale à la probabilité que l'exécution de la tâche consomme plus de ressources qu'il n'y en a de disponibles.

• **Contraintes temporelles $TC(t_{i+1})$ non respectées :**

Les contraintes temporelles sont violées lorsque l'agent $\mathcal{A}g_i$:

- commence l'exécution de la tâche après sa date de début au plus tard ($st > LST_{i+1}$),
- commence l'exécution de la tâche à $st = LST_{i+1}$ mais échoue car les contraintes de précédence ne sont pas respectées,
- ou que l'exécution de la tâche finit après la date de fin au plus tard.

Ce dernier cas se produit si les contraintes de précédence et de ressources sont respectées, que la date de début de la tâche est inférieure à LST_{i+1} mais que la durée de la tâche est telle que $st + \delta_c^{i+1} > LET_{i+1}$. La probabilité d'une telle transition est calculée à partir des probabilités sur les durées d'exécution.

La figure 5.3 récapitule les différentes transitions possibles lorsqu'un agent décide, à partir d'un état s_i, d'exécuter une tâche t_{i+1} à st. Les raisons provoquant le passage par une transition donnée sont mentionnées sur la figure. Ce modèle de transition permet de prendre en considération les différentes contraintes régissant l'exécution des tâches.

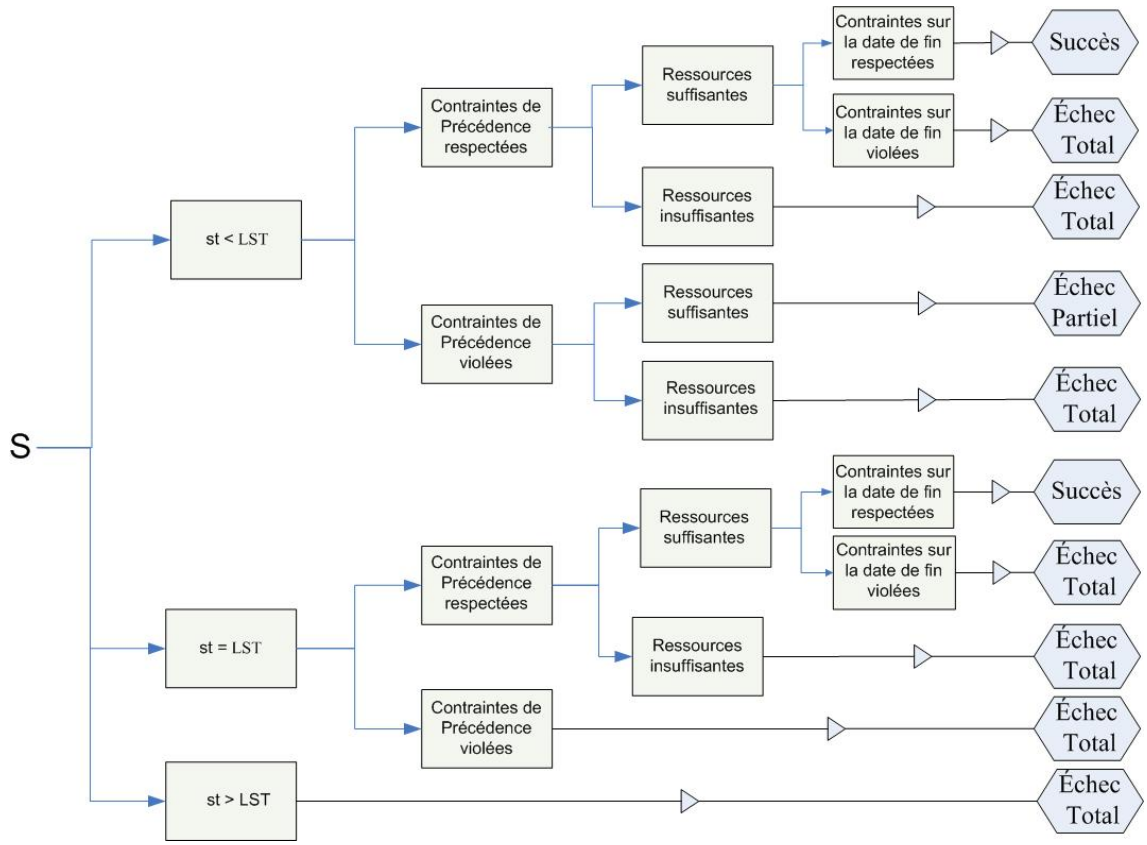


FIG. 5.3 – Modèle de transition et taxonomie d'état

En raison des contraintes de précédence, nous avons pu voir que l'exécution d'une tâche par un agent dépendait des autres agents. Ainsi, les fonctions de transition des autres agents ne sont pas indépendantes les unes des autres. $\mathcal{P}_i$ est donc défini *a priori* comme une fonction $\mathcal{P}_i : \mathcal{S}_i \times \mathcal{A} \times \mathcal{S}_i \rightarrow [0,1]$ et non comme une fonction $\mathcal{P}_i : \mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i \rightarrow [0,1]$ ne dépendant que de l'action exécutée par l'agent.

5.3.4 Fonction de récompense

La fonction de récompense du problème associe à chaque tâche, une récompense. Lorsqu'un agent arrive dans un état « succès », le système reçoit donc la récompense rattachée à la tâche qui vient d'être exécutée.

Lorsqu'un agent arrive dans un état d'échec total, il reçoit une pénalité égale à la somme des récompenses associées aux tâches qui ne pourront être exécutées.

Notons qu'aucune récompense n'est obtenue pour le passage par un état d'échec partiel puisqu'aucune tâche n'est réalisée.

Comme nous l'avons fait remarquer à plusieurs reprises dans cette section, les différentes composantes des MDPs locaux sont définies de façon à modéliser les contraintes sur l'exécution des tâches. Les états de succès représentent des situations dans lesquels les contraintes sont respectées alors que les états d'échec formalisent des situations de violation des contraintes. De même, les actions sont définies de manière à respecter les contraintes temporelles. La fonction de récompense modélise, quant à elle, les violations de contraintes et le succès de l'exécution d'une tâche. Les contraintes sont donc intrinsèquement formalisées dans chacun des MDPs locaux ce qui garantira leur respect lors de la résolution du problème.

5.4 Propriétés des OC-DEC-MDPs

Comme l'a démontré Bernstein [Bernstein *et al.*, 2002], la résolution optimale d'un DEC-POMDP à deux agents ou plus est un problème NEXP. Lors de la présentation des DEC-POMDPs et DEC-MDPs, nous avons introduit un certain nombre de propriétés permettant d'identifier des classes de problèmes de complexité moins importante. C'est pourquoi, afin de justifier la complexité des problèmes que nous traitons, la spécification de leurs propriétés est tout d'abord nécessaire.

5.4.1 Transitions et Observations dépendantes

Du fait des contraintes de précédence, la probabilité qu'un agent passe d'un état s_i à un état s'_i dépend des états et actions des autres agents. Les problèmes abordés sont par conséquent à transitions dépendantes.

Ces problèmes sont de plus à observations dépendantes. Les états des agents étant localement totalement observables, nous pouvons établir une bijection entre l'ensemble des états de l'agent et ses observations. En raison des contraintes de précédence, l'état d'un agent dépend des états des autres agents. Par conséquent, les observations d'un agent dépendent des observations des autres agents.

5.4.2 Indépendance des récompenses

La fonction de récompense peut être décomposée en fonctions de récompense individuelles et indépendantes les unes des autres. En effet, une récompense est octroyée au système à la fin de l'exécution de chaque tâche. Une tâche n'étant réalisée que par un seul agent, les récompenses ne dépendent donc que d'actions individuelles et non des actions jointes exécutées par les agents. Il est alors possible de définir une fonction de récompense pour chaque agent ne dépendant que de l'état et de l'action entreprise par l'agent.

5.4.3 Observabilité

Afin que les états soient totalement observables, il est nécessaire de connaître toutes les valeurs de r et I pouvant être associées à chaque tâche t_i. Nous présenterons dans la section 5.5 deux algorithmes permettant de réaliser ces calculs par propagation dans le graphe de la mission. Grâce au calcul de ces données, nous sommes en mesure de construire, pour chaque agent, un espace d'état localement totalement observable.

Bien que les états des agents soient localement totalement observables, l'état du système reste partiellement observable. Néanmoins, si les agents globalisent leurs connaissances, toutes les observations nécessaires à la prise de décision des agents seront connues d'où une observabilité collective totale. Étant donné que les agents ne peuvent pas communiquer durant l'exécution de leurs tâches, cette observabilité totale ne peut cependant pas être atteinte en pratique.

Les propriétés des problèmes considérés sont résumés dans le tableau 5.1.

	Transition Indépendance	Observation Indépendance	Orienté par un but	Observabilité	
				locale	du système
OC-DEC-MDP	Non	Non	Non	Totale	Partielle

TAB. 5.1 – Propriétés des OC-DEC-MDPs

5.4.4 Complexité

Nous avons montré que les problèmes traités étaient à transitions et observations dépendantes. De plus, le comportement des agents n'est pas orienté vers un but. Les problèmes traités ne possèdent donc pas de propriétés permettant de les identifier comme des classes de DEC-POMDPs connues pour avoir des complexités moins importantes [Goldman et Zilberstein, 2004]. Nous pouvons néanmoins exploiter l'observabilité locale des états et démontrer le théorème suivant :

Théorème 1 *La complexité en temps d'un OC-DEC-MDP est exponentielle en espace d'états.*

Preuve : Les états des agents sont localement totalement observables. Une politique jointe fait donc correspondre à chaque ensemble d'états $\langle \mathcal{S}_1, \cdots, \mathcal{S}_n \rangle$ une action jointe $\mathcal{A} = \langle \mathcal{A}_1, \cdots, \mathcal{A}_n \rangle$. Par conséquent, il existe un nombre de politiques jointes exponentiel en espace d'états.

L'évaluation d'une politique jointe peut être réalisée en temps linéaire à l'aide d'un algorithme de programmation dynamique. Néanmoins, la recherche de la politique optimale nécessite, dans le pire cas, de parcourir tout cet espace et a donc une complexité exponentielle. □

Remarquons que les contraintes du problème influencent la taille de l'espace d'états et donc la taille et le nombre de politiques. En effet, elles restreignent le nombre d'états et d'actions possibles pour chaque agent. Cependant, ces contraintes n'affectent en rien la complexité au pire cas de l'algorithme, le nombre de politiques possibles demeure exponentiel en espace d'états. Les contraintes ne réduisent donc pas la classe de complexité du problème.

5.4.5 Comparaison avec l'existant

La figure 5.4 reprend la classification des approches s'intéressant aux processus décisionnels de Markov décentralisés et place notre travail par rapport à l'existant. Étant donné l'observabilité collective totale des agents nous nous plaçons dans le cadre des DEC-MDPs. Contrairement aux DEC-MDP-Com et COM-MTDP, nos agents n'ont pas la possibilité de communiquer durant la phase d'exécution des tâches (en ligne). Les communications n'ont donc pas à être modélisées. En revanche, nous devons faire face à des problèmes de coordination entre agents du fait de cette absence de communication et de l'observabilité partielle des autres agents.

Comme nous l'avons déjà précisé, nous nous plaçons dans le contexte général des problèmes à transitions et observations dépendantes. Contrairement aux approches traitant des DEC-POMDP-Com [Goldman et Zilberstein, 2003] et TI-DEC-MDP [Becker *et al.*, 2003], nous n'émettons donc pas d'hypothèses particulières sur la structure des fonctions de transition et d'observation.

Les ED-DEC-MDP constituent la classe de problèmes la plus proche de notre travail. En effet, les ED-DEC-MDPs traitent comme nous des problèmes à observations et transitions dépendantes. De plus, ils permettent la prise en compte de dépendances entre les tâches. Des contraintes de précédence sont en effet gérées et représentées à l'aide du formalisme TAEMS.

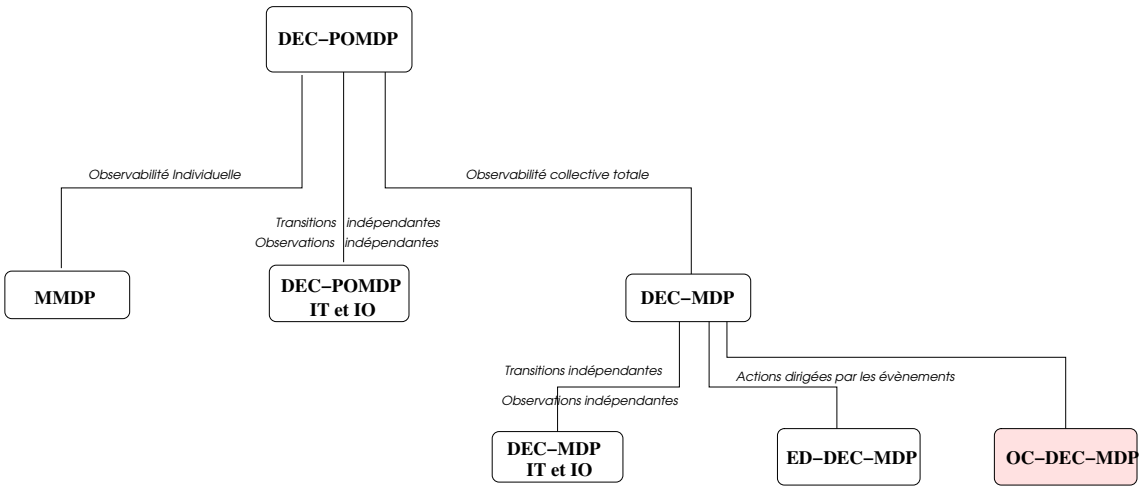


FIG. 5.4 – Mise en relation des OC-DEC-MDPs avec les modèles existants

Les OC-DEC-MDPs et ED-DEC-MDPs sont à notre connaissance, les seuls travaux s'intéressant à la gestion de contraintes dans les DEC-MDPs. De plus, ils permettent, contrairement aux autres approches, la gestion de différentes durées possibles pour l'exécution des tâches. Néanmoins, les ED-DEC-MDPs ne traitent pas des contraintes de ressources bien que le modèle puisse être étendu afin d'en tenir compte.

5.5 Enrichissement des données par propagation

Nous allons clore ce chapitre par la description d'une méthode pour la construction « automatique » d'un OC-DEC-MDP à partir d'une mission $\mathcal{X}$. Nous allons, dans les deux sections qui suivent, décrire plus concrètement comment construire ce modèle à partir d'un graphe de mission donné. Nous commencerons par décrire deux algorithmes permettant d'enrichir les connaissances sur le problème [Beynier et Mouaddib, 2004a, Beynier et Mouaddib, 2004b]. Étant donnée une mission, ces algorithmes procèdent, par propagation des contraintes temporelles et des contraintes de ressources, au raffinement des données sur l'exécution des tâches. Ainsi, nous calculons les intervalles d'exécution possibles de chaque tâche et les taux de ressources disponibles après leur exécution. Ces informations seront ensuite utilisées pour la construction des MDPs locaux.

5.5.1 Propagation temporelle

Étant donnée l'incertitude sur l'exécution des tâches, il existe un grand nombre d'intervalles d'exécution possibles pour chacune d'elles. Nous appellerons intervalle d'exécution, un intervalle I constitué de la date de début de la tâche et de sa date de fin. Rappelons que pour chaque tâche t_i, son intervalle d'exécution I est tel que $I \subset [EST_i, LET_i]$, c'est-à-dire tel que les contraintes temporelles $TC(t_i)$ soient respectées.

Les contraintes de précédence de la tâche doivent également être considérées. En effet, toute tâche ne peut commencer que si ses prédécesseurs ont terminé leur exécution. La date de début d'une tâche t_i dépend donc des dates de fin de ses prédécesseurs.

Parcours des tâches

A partir des contraintes temporelles, de précédence et des différentes durées d'exécution possibles de chaque tâche t_i, nous souhaitons calculer tous les intervalles pouvant correspondre à une fenêtre d'exécution de t_i. Pour ce faire, nous parcourons le graphe de la mission niveau par niveau en considérant chaque tâche de chaque niveau [Beynier et Mouaddib, 2004c].

Le premier niveau L_0 est constitué des racines du graphe c'est-à-dire des tâches n'ayant pas de prédécesseurs. Le second niveau est constitué de tous les successeurs des tâches racines. Le troisième niveau est composé de tous les successeurs des tâches du second niveau dont on a

calculé les intervalles, et des tâches du second niveau dont les intervalles n'ont pu être calculés. De manière plus générale, un niveau L_k contient tous les successeurs des tâches du niveau L_{k-1} dont on a calculé les intervalles, et contient également les tâches du niveau L_{k-1} dont les intervalles n'ont pu être déterminés. En effet, il est possible de déterminer les intervalles d'une tâche $t_i \in L_k$ si et seulement si les intervalles de tous ses prédécesseurs ont été calculés. Or, le fait qu'une tâche t_i appartienne à L_k signifie qu'au moins un de ses prédécesseurs appartient à un niveau inférieur, mais cela ne garantit en aucun cas que les intervalles de tous ses prédécesseurs aient été déterminés. Lorsqu'un niveau L_k est parcouru, le niveau L_{k+1} est initialement vide et est construit au fur et à mesure. Lorsqu'une tâche t_i est envisagée, si les intervalles des prédécesseurs n'ont pas encore été tous définis, alors la tâche sera reconsidérée au niveau suivant auquel elle est ajoutée. Il est inutile de se préoccuper des successeurs de cette tâche et de les ajouter au niveau L_{k+1} puisque un de leurs prédécesseurs y appartient et n'a donc pas encore été traité.

Si tous les intervalles des prédécesseurs de $t_i \in L_k$ ont été déterminés, nous effectuons les calculs pour la tâche : calcul des dates de début, calcul des dates de fin, calcul des intervalles d'exécution. Ensuite, les successeurs de la tâche, sont ajoutés au niveau L_{k+1}.

Dates de début

Les dates de début des tâches à la racine du graphe, sont déterminées à partir de leurs contraintes temporelles et de la date à laquelle le système « se réveille », désignée par $start\_time$. Ainsi, nous obtenons :

$$st(root) = max(\{EST_{root}, start\_time\})$$

Considérons toute autre tâche $t_i \notin L_0$. Ses dates de début appartiennent à l'intervalle $[EST_i, LST_i]$ où LST_i désigne la date de début au plus tard de la tâche (Latest Start Time) et est définie par : $LST_i = LET_i - \delta_c^{i,min}$ où $\delta_c^{i,min}$ est la plus petite durée de la tâche t_i.

L'ensemble de ces dates peut être restreint en considérant les dates de fin des prédécesseurs. En effet, il est possible de déterminer une date de fin au plus tôt (FET, First End Time) des prédécesseurs avant laquelle la tâche ne pourra commencer :

$$FET_i = \max\Big(\bigcup_{pred \in Pred(t_i)} (\min(ET(pred)))\Big)$$

Le maximum entre cette date et la date de début au plus tôt de la tâche permet de déterminer la première date de début possible de la tâche. Nous la noterons LB_i (Lower Bound) :

$$LB_i = \max(\{EST_i, FET_i\})$$

L'ensemble des dates de début possibles est alors déduit à partir des dates de fin des prédécesseurs. Nous ne considérons pas le fait qu'un agent puisse être dans l'obligation de respecter

Algorithme 3 : Calcul des intervalles d'exécution

Entrées : un problème $\mathcal{X}$

1 $Niv \leftarrow root$ // tâches du niveau courant
2 $NivSuiv \leftarrow \emptyset$ // tâches à considérer au niveau suivant
3 **répéter**
4 **pour tous les** $t_i \in Niv$ **faire**
5 **si** *les intervalles de tous les prédécesseurs de* t_i *ont été calculés* **alors**
 // Calcul de l'ensembles des dates de début de t_i
6 **si** $t_i \in root$ **alors** /* la tâche est une racine du graphe */
7 $st(t_i) = max(\{EST(t_i), start\_time\}) = LB_{t_i} = UB_{t_i}$
8 $ET(t_i) = \{st(t_i) + \delta_c^i \leq LET, \forall \delta_c^{t_i}\}$
9 **sinon**
10 $LB_i = \max(\{EST_i, FET_i\})$
11 $OST_i = \{et \in ET(t_i), et > FETAP_i\}$
12 $ST(t_i) = \{st \in FETAP_i \bigcup OST_i\}$ tel que $EST_i \leq st \leq LST_i$
13 $UB_i = max(ST(t_i))$
14 **fin**
 // Calcul de l'ensemble des dates de fin d'exécution de t_i
15 **pour tous les** $st \in ST(t_i)$ **faire**
16 **pour tous les** δ_c^i **faire**
17 $ET(t_i) = \{st + \delta_c^i LET_i\}$
18 **fin**
19 **fin**
 // Calcul des intervalles d'exécution de t_i
20 **pour tous les** $st \in ST(t_i)$ **faire**
21 **pour tous les** δ_c^i **faire**
22 $I = [st, st + \delta_c^i]$
23 **fin**
24 **fin**
25 $NivSuiv \leftarrow NivSuiv \cup Succ(t_i)$
26 **sinon**
 // les prédécesseurs n'ont pas tous été considérés
27 $NivSuiv \leftarrow NivSuiv \cup t_i$
28 **fin**
29 **fin**
30 $Niv \leftarrow NivSuiv$
31 $NivSuiv \leftarrow \emptyset$
32 **jusqu'à** $Niv = \emptyset$;

Sorties : $\mathcal{X}$ enrichi de l'ensemble des intervalles d'exécution de chaque tâche $t_i \in \mathcal{T}$

un délai donné entre l'exécution de deux tâches. Par exemple, un agent doit peindre une porte puis attendre 3 heures avant de passer la seconde couche[32]. De ce fait, dès qu'un agent finit l'exécution d'une tâche, il peut commencer la tâche suivante. La seule chose qui peut retarder l'exécution de la tâche suivante est le fait que les prédécesseurs n'aient pas terminé.

Nous désignerons par OST_i (Other Start Time) l'ensemble des dates de fin de prédécesseurs et par ST l'ensemble des dates de début possibles de la tâche t_i. Nous pouvons affirmer que $st \in ST(t_i)$ (st est une date de début possible pour la tâche t_i) si et seulement si $st \geq LB_i$ et $st \in OST_i$. Désignons par t_{i-1} la dernière tâche exécutée par l'agent devant réaliser t_i. t_{i-1} est un prédécesseur indirect ou direct de t_i, donc $st \leq max(OST_i)$. Raisonnons alors par l'absurde et supposons que l'agent chargé de la tâche t_i commence son exécution à $st > LB_i$ tel que $st \notin OST_i$. $st \leq max(OST_i)$. Si l'agent commence la tâche t_i à st, il risque de perdre du temps ou des ressources pour rien. En effet, si les prédécesseurs ont terminé à st, comme st n'est pas une date de fin des prédécesseurs, cela signifie que les prédécesseurs avaient déjà terminé avant st. Le but des agents étant de maximiser la somme des récompenses obtenues, ils n'ont aucun intérêt à rester à ne rien faire sans raison, et par conséquent à commencer à st. Si les prédécesseurs n'ont pas terminé à st, l'agent aura essayé d'exécuter la tâche et donc aura consommé des ressources alors que les prédécesseurs finissent obligatoirement avant st ou après st. Dans ce cas, les ressources sont consommées sans résultat. Afin de maximiser son espérance de gain, l'agent sélectionnera donc nécessairement une date de début st correspondant à une date de fin des prédécesseurs, d'où $st \in OST_i$.

L'ensemble des dates de début d'une tâche t_i est donc constitué des dates de fin de ses prédécesseurs postérieures à LB_{t_i}. D'où :

$$OST_i = \bigcup_{pred \in Pred(t_i)} \{et \in ET(pred), et > LB_i\}$$

$$ST(t_i) = \{LB_i \bigcup OST_i\}$$

Nous noterons par UB_i (Upper Bound) la dernière date de début possible de la tâche t_i. Elle est définie par :

$$UB_i = max(ST(t_i))$$

Nous tenons à préciser que « dernière date de début possible » et « date de début au plus tard » sont à différencier. La première correspond à une date de début effective de la tâche alors que la seconde traduit des contraintes temporelles. Il en est de même pour « première date de début » et « date de début au plus tôt ».

[32] Il est cependant possible d'adapter le calcul des dates de début pour considérer de tels cas.

Dates de fin

A partir des dates de début possibles et de l'ensemble des différentes durées de la tâche, nous pouvons déduire l'ensemble des dates de fin possibles défini par :

$$ET(t_i) = \{st + \delta_c^i < LET_{t_i}\}$$

Intervalles

L'ensemble des intervalles d'exécution de la tâche sont obtenus de la même façon. Tout intervalle d'exécution I d'une tâche t_i est tel que :

$$I = [st \in ST(t_i), st + \delta_c^i]$$

Afin de calculer l'ensembles des intervalles d'exécution possibles, il n'est donc pas nécessaire d'avoir préalablement défini l'ensembles des dates de fin possibles.

L'algorithme permettant ces calculs est décrit par l'algorithme 3. Il correspond à une propagation en avant des contraintes temporelles dans le graphe de la mission. Un exemple de calcul des intervalles est présenté sur la figure 5.5. Les durées d'exécution ainsi que les contraintes temporelles mentionnées pour chaque tâche permettent de déduire les intervalles d'exécution des tâches. Notons que l'allocation des tâches sur les agents ne joue aucun rôle dans ces calculs. Les calculs sont tout d'abord réalisés pour la tâche « Se rendre sur le site A ». Puis, la tâche « Prendre une photo du site A » est considérée. Enfin, le dernier niveau parcouru est composé des tâches « Entrer sur le site A » et « Creuser sur le site A ».

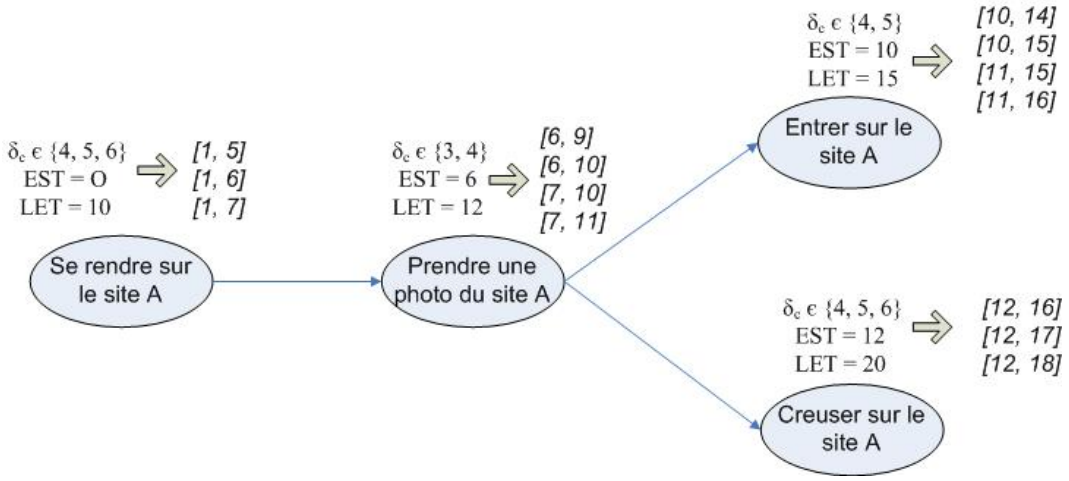


FIG. 5.5 – Exemple de calcul des intervalles d'exécution

5.5.2 Propagation des ressources

Afin de construire l'espace d'états des agents, il est également nécessaire de calculer l'ensemble des taux de ressources disponibles après l'exécution de chaque tâche. Pour ce faire, nous avons développé un algorithme similaire à celui qui vient d'être présenté pour le calcul des intervalles. Cet algorithme, décrit par l'algorithme 4, réalise une propagation des consommations de ressources dans le graphe de la mission. Ce dernier est une fois encore découpé en niveaux parcourus les uns après les autres.

Soit t_i la tâche pour laquelle nous souhaitons calculer les taux de ressources disponibles. Cette tâche est réalisée par un agent Ag_j. Nous désignerons par t_{i-1} la dernière tâche effectuée par l'agent avant t_i.

Nous considérons des ressources non partagées entre les agents, par exemple de l'énergie. Les taux de ressources disponibles avant l'exécution de t_i correspondent donc aux taux de ressources disponibles après l'exécution de t_{i-1}. Étant donnée la procédure de parcours du graphe de la mission, lorsque la tâche t_i est considérée, nous pouvons affirmer que les taux de ressources de la tâche t_{i-1} ont déjà été calculés. Ainsi nous connaissons les taux de ressources r_{i-1} disponibles après l'exécution de t_{i-1}, et par conséquent les taux de ressources possibles avant l'exécution de t_i. Les taux de ressources disponibles après l'exécution de t_i sont obtenus en soustrayant à chaque r_{i-1}, les consommations de ressources $\Delta r'$ des échecs partiels de t_i et la consommation de ressources Δ_r^i de son exécution.

Le principe général de cet algorithme étant identique à celui du calcul des intervalles d'exécution, il est permis de fusionner les deux algorithmes. A partir d'un seul parcours de graphe il est alors possible de calculer, pour chaque tâche t_i, ses dates de début, ses dates de fin, ses intervalles d'exécution et les taux de ressources disponibles après son exécution. A partir de l'ensemble de ces données, nous obtenons directement les états associés à t_i.

5.6 Construction d'un OC-DEC-MDP

Les deux algorithmes qui précèdent permettent d'augmenter nos connaissances sur le problème et plus particulièrement sur les situations d'exécution des tâches. Nous allons à présent décrire comment ces algorithmes s'inscrivent dans le processus complet de construction des OC-DEC-MDPs.

La première étape de cette construction consiste en l'enrichissement des données du problème par propagation des contraintes. Afin de construire les MDPs locaux il en effet nécessaire de déduire certaines informations à partir des données du problème. Pour établir les espaces d'états

Algorithme 4 : Calcul des taux de ressources

Entrées : un problème $\mathcal{X}$

$Niv \leftarrow root$ // tâches du niveau courant
$NivSuiv \leftarrow \emptyset$ // tâches à considérer au niveau suivant
répéter
  pour tous les $t_i \in Niv$ **faire**
    si *les taux de ressources de tous les prédécesseurs de* t_i *ont été calculés* **alors**
      // Calcul des taux de ressources de t_i
      si $t_i \in root$ **alors** /* la tâche est une racine du graphe */
        pour tous les Δ_r^i **faire**
          $res = R_{ini} - \Delta_r^i \; R(t_i) \rightarrow R(t_i) \cup res$
        fin
      sinon
        $R(t_i) \rightarrow \emptyset$
        Soit t_{i-1} la tâche exécutée par l'agent juste avant t_i
        pour tous les $r_{i-1} \in R(t_{i-1})$ **faire**
          // ressources disponibles après l'exécution de t_{i-1}
          pour tous les $k \in [0, |ST(t_i)|]$ **faire** /* nb. échecs partiels */
            pour tous les Δ'_r **faire** /* consommation d'1 échec partiel */
              $r' = r_{i-1} - k \times \Delta'_r$
              si $r' > 0$ **alors**
                pour tous les Δ_r^i **faire**
                  $res = r' - \Delta_r^i$
                  $R(t_i) \rightarrow R(t_i) \cup res$
                fin
              fin
            fin
          fin
        fin
        $NivSuiv \leftarrow NivSuiv \cup Succ(t_i)$
      sinon
        // les prédécesseurs n'ont pas tous été considérés
        $NivSuiv \leftarrow NivSuiv \cup t$
      fin
  fin
  $Niv \leftarrow NivSuiv$
  $NivSuiv \leftarrow \emptyset$
jusqu'à $Niv = \emptyset$;

Sorties : $\mathcal{X}$ enrichi de l'ensemble des taux de ressources pouvant être disponibles avant l'exécution de chaque tâche $t_i \in \mathcal{T}$

et d'actions, il est en effet nécessaire de déterminer les intervalles d'exécution et les taux de ressources possibles pour chaque tâche. Les probabilités de chacun de ces intervalles et taux de ressources sont également calculées, elles sont utilisées pour définir la fonction de transition. Nous détaillerons le calcul de ces probabilités dans la partie suivante. Ces opérations permettent ainsi d'obtenir les informations nécessaires à la construction de l'espace d'états, l'espace d'actions et au calcul de la fonction de transition.

A partir du graphe de la mission, un graphe de tâches peut être déduit pour chaque agent. Il ordonne les tâches que l'agent doit accomplir et établit une trame pour la construction du MDP local de l'agent. La figure 5.6 présente les graphes des tâches des agents déduits du graphe de mission décrit précédemment. Les contraintes de précédence entre les agents conduisent à des dépendances entres les graphes de tâches et par conséquent entre les MDPs locaux. Elles sont mentionnées, sur la figure, par des flèches en pointillés.

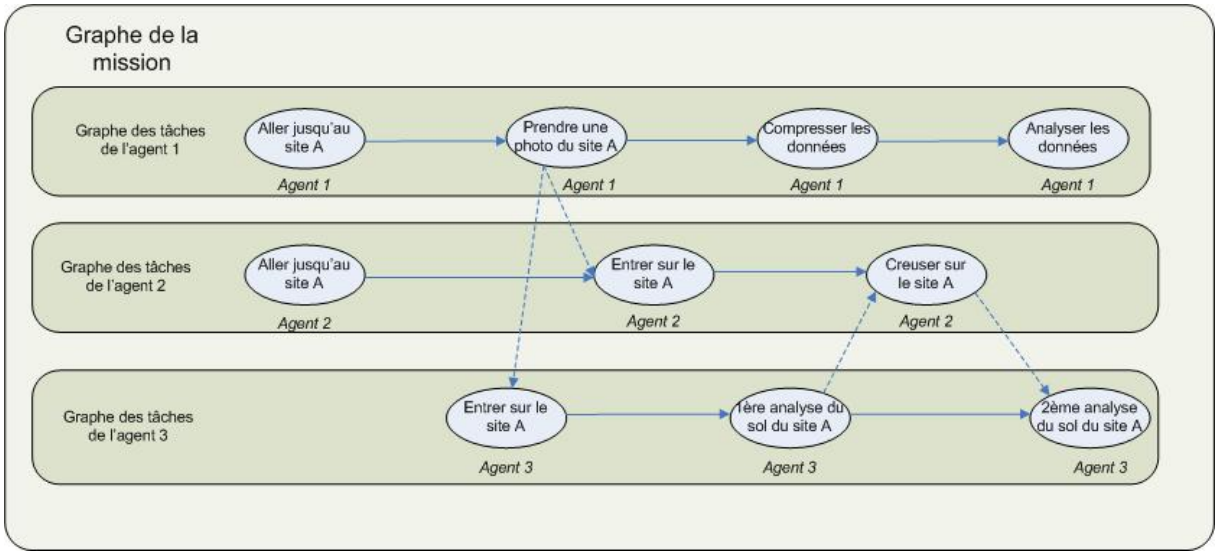


FIG. 5.6 – Relations entre le graphe de la mission et les graphes des tâches des agents

Une fois la propagation des contraintes et le calcul des probabilités réalisés, les MDPs locaux sont déterminés en suivant la structure des graphes de tâches des agents. La construction d'un MDP local passe par la définition :

- d'un espace d'états associant à chaque tâche un ensemble d'états
- d'un ensemble d'actions déterminé à partir des dates de début possibles des tâches
- d'une fonction de transition déduite à partir des probabilités des intervalles et des taux de

ressources

– d'une fonction de récompense.

La figure 5.7 décrit la correspondance entre les graphes de tâches et les espaces d'états des MDPs locaux. A chaque tâche est associé un ensemble d'états construit à partir des intervalles et taux de ressources calculés auparavant. Sur cette figure 5.7, chaque point représente un état et les flèches décrivent des transitions entre états. Elles relient tout état associé à une tâche t_i à un état d'échec partiel associé à cette même tâche, à un état succès lié à la tâche suivante t_{i+1} ou bien à l'état d'échec total de t_{i+1} (nœuds en noir). Les nœuds sans successeurs correspondent à des états terminaux.

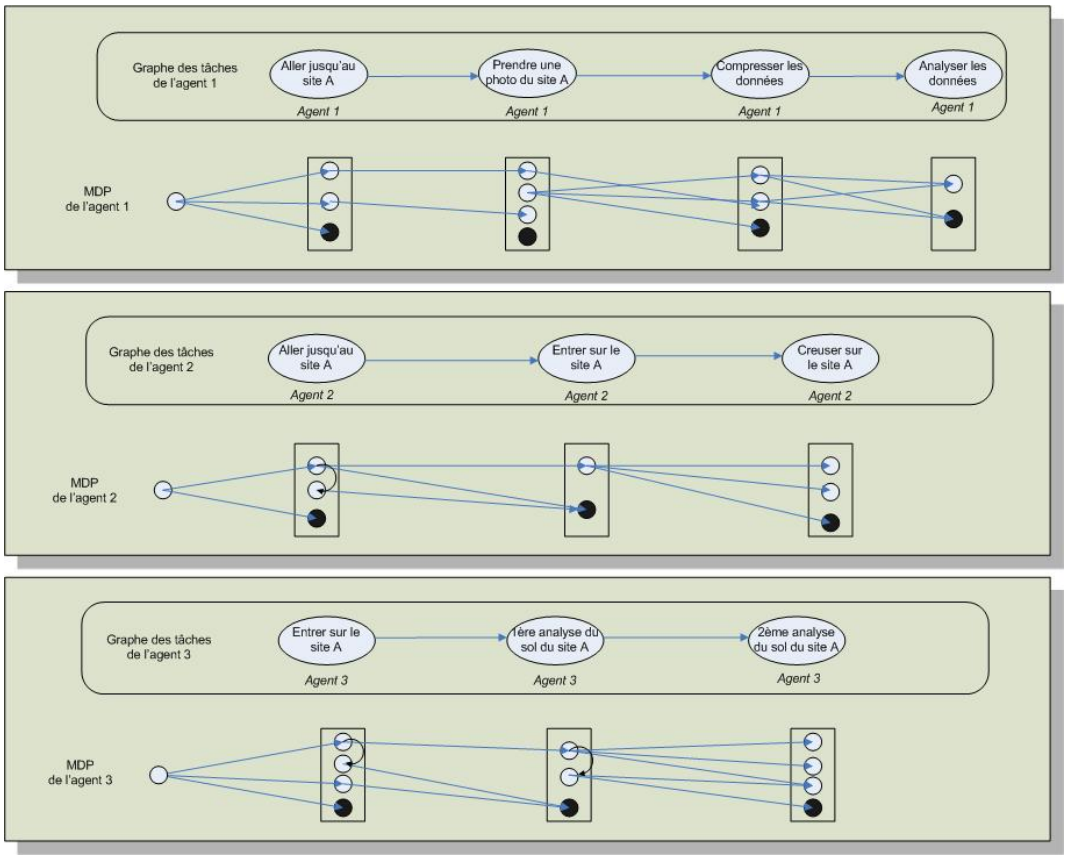


FIG. 5.7 – Relations entre les graphes des tâches et les MDPs des agents

Pour chaque tâche, nous pouvons déduire l'ensemble des états et des actions qui lui sont associés. La matrice de transition décrivant les probabilités des transitions entre les états, est obtenue par combinaison des probabilités des intervalles et des taux de ressources.

Conclusion

Dans ce chapitre, nous nous sommes intéressée à la modélisation d'une mission sous forme de processus décisionnels de Markov décentralisés. La gestion des contraintes et des incertitudes sur les durées d'exécution des tâches a tout d'abord nécessité l'amélioration de la modélisation du temps et des actions réalisée usuellement dans les modèles markoviens.

Nous avons alors montré que la prise en compte, dans les DEC-POMDPs, des incertitudes sur les durées, menait à des modèles de taille très importante et par conséquent très difficile voire impossible à résoudre. Nous avons alors choisi de décomposer le problème de planification multi-agent en un ensemble de MDPs représentant chacun le problème décisionnel d'un agent. Nous avons décrit comment définir chacun de ces MDPs tout en modélisant les différents types de contraintes. Une méthode pour la construction « automatique » des OC-DEC-MDPs a notamment été présentée.

L'exposé des différentes composantes de chaque MDP a permis d'identifer les propriétés des problèmes considérés et de démontrer leur importante complexité. Nous avons également souligné l'existence de dépendances entre les agents traduites par des dépendances entre les MDPs. Le fait que l'exécution des actions d'un agent influence les autres agents et nécessite la mise en place d'un processus de coopération, suggère que les MDPs ne puissent être résolus de façon totalement indépendante.

Conclusion de la partie II

Cette seconde partie nous a permis de définir clairement les problèmes auxquels nous nous intéressons et d'en proposer une modélisation.

Le sujet de notre thèse s'inscrit dans le domaine de la prise de décision pour des colonies de robots autonomes. Afin de motiver les propriétés des problèmes que nous souhaitons résoudre, nous avons tout d'abord décrit différentes applications possibles de notre travail. Nous nous sommes étendue sur les caractéristiques des robots explorateurs et avons montré que d'autres applications des robots mobiles telles que les robots sauveteurs, les robots dockers ou les constellations de satellites, présentaient de grandes similarités.

Nous avons pu mettre en évidence le fait que toutes ces applications consistent en un système multi-agent coopératif dans lequel les agents doivent décider de façon autonome comment agir. Nous avons alors défini par le terme de mission l'ensemble des agents et des tâches constituant les problèmes considérés. Il a de plus été montré que toutes ces applications partageaient un certain nombre de particularités :

- les agents ne peuvent pas communiquer durant l'exécution des tâches,
- les agents possèdent des capacités de calcul limitées,
- l'exécution des tâches est incertaine (en temps et en consommation de ressources),
- différents types de contraintes doivent être respectés (contraintes temporelles, de précédence, de ressources).

Nous nous sommes ensuite intéressée au processus de prise de décision développé par les agents en vue d'exécuter les tâches de la mission. En raison des capacités de calcul limitées et de l'impossibilité de communiquer durant l'exécution, nous nous sommes orientée vers un processus de planification hors-ligne de l'exécution des tâches.

Nous avons adopté, en première partie de cette thèse, le point de vue de la rationalité afin de définir l'intelligence d'un agent. Nous avions également présenté le principe de maximisation

de l'utilité espérée comme un moyen de garantir la rationalité d'un comportement. Le recours à ce principe, la décentralisation du contrôle, ainsi que la nécessaire gestion de l'incertitude, nous ont conduit à modéliser les missions sous forme de DEC-MDPs.

Le second chapitre de cette partie a eu pour objectif d'expliciter cette modélisation. Nous avons montré que la gestion des incertitudes sur les durées d'exécution, par le modèle usuel des DEC-MDPs, conduit à des représentations de tailles conséquentes. Étant données les difficultés exposées en première partie sur la résolution des DEC-MDPs de petite taille, il nous a semblé inenvisageable d'utiliser une telle modélisation. Le fait que nous souhaitions résoudre des problèmes composés de plus d'une centaine de tâches a d'autant plus renforcé cette idée.

Nous avons ainsi été amenée à introduire une nouvelle classe de DEC-MDPs, les OC-DEC-MDPs. Nous avons procédé à une décomposition du problème en un ensemble de MDPs représentant chacun le problème décisionnel d'un agent. Les différentes composantes de ces MDPs ont été définies de manière à tenir compte des différentes contraintes régissant l'exécution des tâches. Il a alors été montré que les problèmes que nous traitons sont à observations dépendantes, transitions dépendantes et récompenses indépendantes. Chaque agent possède par ailleurs, une observabilité totale de son état. Nous avons ainsi pu prouver que la complexité des problèmes traités était exponentielle en espace d'états. Ces propriétés ont également mis en évidence des dépendances entre les MDPs dues principalement aux contraintes temporelles et de précédence.

Cette seconde partie a donc permis d'établir une modélisation précise des problèmes auxquels nous nous intéressons. Nous allons à présent nous intéresser à leur résolution. La modélisation des contraintes par chaque composante des OC-DEC-MDPs assure leur prise en compte lors de la planification. Les dépendances entre les MDPs locaux suggèrent par ailleurs qu'une résolution indépendante de chacun des MDPs n'est pas adaptée puisqu'elle conduirait à un comportement non coopératif. Nous allons donc chercher à mettre en place un processus de coordination des politiques des agents leur permettant de tenir compte de l'influence de leurs décisions sur les autres agents. La méthode de résolution développée devra par ailleurs être de faible complexité afin de permettre la gestion de tailles de problèmes importantes.

Troisième partie

Résolution d'un OC-DEC-MDP

Introduction de la partie III

Les pages qui précédent nous ont permis de traiter une partie de la problématique de notre travail : la modélisation, par un processus décisionnels de Markov décentralisé, de contraintes portant sur l'exécution des tâches. Nous avons en effet proposé un modèle, l'OC-DEC-MDP, permettant de rendre compte des différents types de contraintes (temporelles ou de ressources) précédemment répertoriés.

Se pose à présent la question de la résolution des OC-DEC-MDPs. Nous souhaitons procéder à la planification de l'exécution des tâches, en vue d'obtenir un ensemble de politiques pouvant ensuite être utilisé par les agents afin de prendre des décisions. Nous allons, dans cette troisième partie, proposer un algorithme efficace permettant de réaliser cette planification de sorte que les agents puissent exécuter leurs tâches sans communiquer, tout en minimisant les calculs nécessaires à la prise de décision.

Afin de développer un tel algorithme, nous avons précédemment procédé à la décomposition du problème en un ensemble de MDPs. Nous commencerons par revenir sur la définition de ces MDPs et traiterons des difficultés liées à la mise en place de fonctions de transition propres à chacun d'eux. Une fois la construction des MDPs achevée, nous pourrons nous atteler à leur résolution.

Pour coordonner les comportements des agents, nous introduirons la notion de coût occasionné. Conjugué à la mesure d'utilité espérée, celui-ci permettra de décider de la politique d'un agent tout en tenant compte de l'impact de chaque action, sur les autres agents. Nous présenterons alors un premier algorithme permettant d'appliquer, à tout état de tout agent, un principe de planification basé sur un compromis entre utilité espéré et coût occasionné. Après avoir discuté de la complexité et de l'optimalité de cet algorithme, nous en décrirons une version itérative, développée afin d'améliorer les résultats obtenus, tout en conservant l'efficacité de la planification.

Chapitre 6

Décomposition du modèle de transition

Dans la partie précédente de ce document, nous avons modélisé les problèmes de planification multi-agent qui nous sont posés sous forme d'OC-DEC-MDPs. Nous avons alors procédé à une décomposition de ces problèmes en un ensemble de MDPs locaux dont nous avons défini chacune des composantes. Les transitions possibles entre les états d'un même MDP ont été décrites en laissant cependant de côté le calcul des probabilités de transition. Nous allons à présent nous intéresser à ces calculs et ainsi proposer un algorithme permettant de déterminer les probabilités nécessaires à la définition d'une fonction de transition pour chaque MDP.

Nous avons précédemment montré que les problèmes abordés étaient, en raison des contraintes de précédence entre les tâches, à transitions dépendantes. La fonction de transition de ces problèmes ne présente donc pas une structure permettant naturellement de la décomposer en fonctions de transition indépendantes les unes des autres et propres à chaque agent. Nous allons, dans ce chapitre, poser les hypothèses permettant d'aboutir à une telle décomposition. Nous montrerons alors comment construire, pour chaque agent, une fonction de transition individuelle permettant à chaque MDP d'être résolu indépendamment des autres[33].

Nous commencerons par poser les hypothèses nécessaires à la décomposition de la fonction de transition. Nous identifierons ensuite les différentes probabilités intervenant dans le calcul des probabilités de transition et nous mettrons en évidence leurs dépendances. Nous proposerons alors un ensemble d'équations ainsi qu'un algorithme permettant leur calcul. Enfin, nous détaillerons comment, à partir de ces probabilités, déduire les fonctions de transition individuelles.

6.1 Principe de décomposition de la fonction de transition

Nous avons présenté au chapitre précédent les différents types de transitions entre états d'un même agent et avons décrit les différents facteurs influençant le passage d'un état s_i à un état s'_i.

[33] Comme nous l'expliquerons au prochain chapitre, une telle résolution ne mène cependant pas à un comportement coopératif de la part des agents.

Nous avons ainsi mis en évidence l'influence, sur les transitions en relation avec l'exécution d'une tâche t_{i+1}, des dates de fin des prédécesseurs de t_{i+1}, de la politique d'exécution de t_{i+1}, de sa consommation de ressources et de sa durée d'exécution. Afin d'établir la fonction de transition individuelle d'un agent, nous avons par conséquent besoin de connaître les probabilités sur :

- les dates de fin des tâches,
- les dates de début des tâches,
- les consommations de ressources,
- les durées d'exécution.

Les deux derniers types de probabilités sont fournis par les données du problème. En revanche, les probabilités sur les dates de début et de fin d'exécution doivent être calculées.

6.1.1 Dépendances entre les fonctions de transition des agents

Les dépendances entre les fonctions de transition des agents sont le résultat des contraintes de précédence existant entre les tâches. En effet, comme nous l'avons expliqué précédemment, une tâche ne peut commencer que si l'exécution de ses prédécesseurs est terminée. La probabilité qu'un agent puisse commencer l'exécution d'une tâche t_{i+1} dépend donc de la probabilité que les prédécesseurs de t_{i+1} aient terminé. Si l'agent connaît les probabilités sur les dates de fin de ses prédécesseurs, il est en mesure de calculer ses propres probabilités de transition, indépendamment des états et des actions des autres agents. Il est alors possible de construire une fonction de transition individuelle $\mathcal{P}_i$ telle que $\mathcal{P}_i : \mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i \rightarrow [0,1]$ ($\mathcal{A}_i$ étant une action individuelle). En revanche, si l'agent ne connaît pas les probabilités sur les dates de fin de ses prédécesseurs, il lui est nécessaire de savoir quelles actions sont entreprises par les autres agents afin de déduire la probabilité qu'il passe d'un état s_i à un état s'_i. La fonction de transition n'est alors pas indépendante des autres agents puisqu'elle se définit de la façon suivante : $\mathcal{P}_i : \mathcal{S}_i \times \mathcal{A} \times \mathcal{S}_i \rightarrow [0,1]$ ($\mathcal{A}$ étant une action jointe).

Examinons les connaissances nécessaires pour calculer les probabilités sur les dates de fin des prédécesseurs $Pred(t_{i+1})$ d'une tâche t_{i+1}. Nous pouvons constater que les dates de fin des tâches t_j appartenant à $Pred(t_{i+1})$ dépendent des dates de début de ces mêmes tâches et de leurs durées d'exécution. La date de début d'une tâche t_j repose quant à elle sur les dates de fin de ses prédécesseurs (prédécesseurs des prédécesseurs de t_{i+1}). La date de début de t_{i+1} dépend donc de ses prédécesseurs $Pred(t_{i+1})$ qui dépendent de l'ensemble des prédécesseurs défini par : $\bigcup_{t_j \in Pred(t_{i+1})} Pred(t_j)$. Par récurrence, nous pouvons ainsi « remonter » jusqu'aux tâches racines du graphe de la mission. Les probabilités sur les dates de début de t_{i+1} dépendent en effet indirectement des probabilités sur les dates de début et de fin de toutes les tâches en amont

de t_{i+1} dans le graphe de la mission. Prenons l'exemple de la mission décrite sur la figure 6.1. La date de début de la tâche E dépend de la date de fin de la tâche D et donc de la date de début de D. Cette dernière est influencée par les dates de fin de B et C qui sont respectivement dépendantes des dates de début de B et C. Ces dernières dépendent de la date de fin de A qui est liée à la date de début de A, elle même déterminée par la date à laquelle le système se réveille et les contraintes temporelles de A.

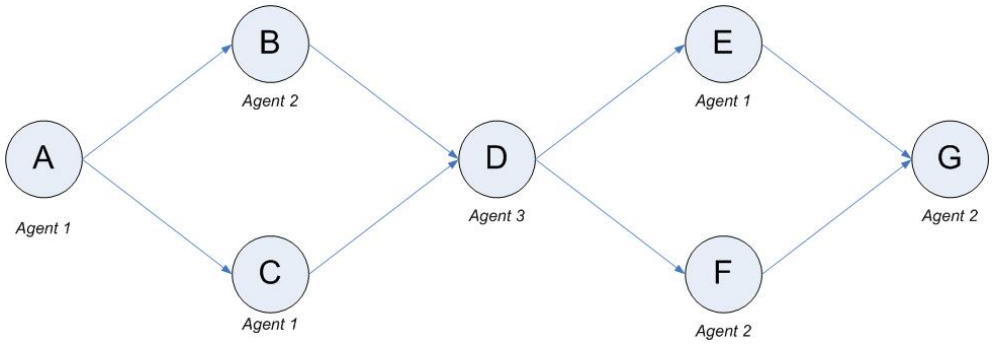


FIG. 6.1 – Exemple de mission

La date de début d'une tâche t_{i+1} ne dépend cependant pas seulement des dates de fin de ses prédécesseurs. Elle est également conditionnée par la politique de l'agent exécutant t_{i+1}. Si les prédécesseurs de D terminent à 12h, l'agent 3 peut par exemple décider de commencer à 12h, à 13h ou bien à 14h. La date de début, et par conséquent les dates de fin, de la tâche D sont alors dépendantes de la politique de l'agent 3.

Afin qu'un agent $\mathcal{A}g_i$ puisse calculer la probabilité de passer d'un état s_i à un état s'_i, il doit donc avoir connaissance des politiques des autres agents. En effet, en raison des relations de dépendances entre les tâches, les probabilités sur les dates de fin des prédécesseurs d'une tâche t_{i+1} dépendent des politiques d'exécution des tâches en amont de t_{i+1}. Ces politiques doivent donc être connues de l'agent $\mathcal{A}g_i$.

Prenons l'exemple décrit sur la figure 6.1. Afin que l'agent 3 détermine ses probabilités de transition lors de l'exécution de D, les probabilités sur les dates de fin de B et C doivent être connues. Pour ce faire, les probabilités sur les dates de début de B et C doivent être calculées et les probabilités sur les dates de fin et de début de A doivent être déterminées. Afin de réaliser ces calculs, l'agent doit donc connaître les politiques pour l'exécution des tâches A, B et C.

6.1.2 Hypothèse pour la décomposition

Supposons que nous connaissions la politique de chaque agent. Nous pouvons alors déterminer les dates de début des tâches « racines » du graphe de la mission. En utilisant les probabilités sur les durées d'exécution de ces tâches, nous sommes également en mesure de déduire les probabilités sur leurs dates de fin. Nous pouvons alors considérer les successeurs des tâches « racines ». Soit t_{i+1} l'une de ces tâches et $\mathcal{A}g_i$ l'agent chargé de l'exécuter. A partir des probabilités sur les dates de fin des tâches « racines », et en considérant la politique de l'agent $\mathcal{A}g_i$, nous pouvons déduire les probabilités sur les dates de début de t_{i+1}. En utilisant, les probabilités sur les durées de t_{i+1}, les probabilités sur les dates de fin de la tâche peuvent également être calculées. Ceci étant réalisable pour tous les successeurs t_{suc} des tâches « racines », il est possible de calculer les probabilités sur les dates de fin de toutes les tâches t_{suc} et d'en déduire les probabilités sur les dates de début et de fin des successeurs des tâches t_{suc}. En réitérant ces opérations, nous pouvons déterminer les probabilités sur les dates de début et de fin de toutes les tâches du graphe de la mission.

Toujours sur l'exemple de la figure 6.1, à partir de la date de réveil du système et des contraintes temporelles sur la tâche A, nous pouvons déduire la date de début de A. Puis, en nous appuyant sur les durées d'exécution de A, les probabilités sur les dates de fin de A sont calculables. Une fois ces dernières connues, nous sommes en mesure, à l'aide des politiques des agents, de déterminer les probabilités sur les dates de début des successeurs de A, c'est-à-dire les tâches B et C. Nous pouvons ensuite déterminer les probabilités sur les dates de fin de B et C et par conséquent les probabilités sur les dates de début de D. En propageant ces calculs, nous obtenons les probabilités sur les dates de début et de fin de toutes les tâches du graphe de la mission.

Ainsi, si un agent connaît les politiques de tous les autres agents, il peut déduire les probabilités sur les dates de début et de fin de toutes les tâches de la mission. Il est plus particulièrement capable de calculer les probabilités sur les dates de fin des prédécesseurs des tâches qu'il doit exécuter. Ces politiques lui fournissent donc les informations nécessaires au calcul des probabilités intervenant dans la définition de sa fonction de transition propre : l'agent peut déduire la probabilité qu'il passe d'un état s_i à un état s'_i en réalisant l'action a_i quels que soient s_i, s'_i et a_i. La décomposition de la fonction de transition du problème en fonctions de transition individuelles $\mathcal{P}_i : \mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i \rightarrow [0, 1]$ est par conséquent possible.

6.2 Représentation des dépendances de transition

Afin de permettre le calcul des fonctions de transition individuelles, nous ferons l'hypothèse que les politiques des agents sont connues. Nous considérerons un ensemble de politiques ini-

tialement fixé et composé d'une politique pour chaque agent. A partir de cet ensemble, nous calculerons les probabilités sur les dates de début et de fin de chaque tâche et nous en déduirons la fonction de transition de chaque MDP constituant l'OC-DEC-MDP.

Nous mettrons tout d'abord en évidence, à l'aide des réseaux bayésiens, les relations entre ces probabilités. Nous détaillerons ensuite les équations et l'algorithme permettant leur calcul.

6.2.1 Introduction aux réseaux Bayésiens

Les réseaux bayésiens fournissent un formalisme probabiliste permettant la représentation concise des connaissances que nous avons d'un système. Cette représentation est utilisée afin de prévoir, contrôler, analyser ou simuler le comportement d'un système en vue d'inférer de nouvelles connaissances, de diagnostiquer les causes d'un phénomène ou de prendre des décisions[34].

Un réseau bayésien est un graphe orienté acyclique dans lequel chaque nœud représente une variable aléatoire (discrète ou continue) et où les arcs spécifient les influences entre les variables. L'ensemble des nœuds et des arcs du graphe permet de représenter de manière qualitative les dépendances entres les variables. Afin de représenter l'aspect quantitatif des dépendances, une distribution de probabilités conditionnelles (également appelée table de probabilités conditionnelles) est associée à chaque nœud. Elle décrit l'effet des parents du nœud sur ce dernier.

Illustrons nos propos sur un exemple. John et Marie ont fixé une réunion de travail à 9h. Marie vient en voiture et John en train. Il arrive souvent que John n'entende pas son réveil et manque son train. Il peut par ailleurs être retardé en raison des grèves de train. Marie, quant à elle, part toujours à l'heure de chez elle. En cas de verglas, elle met cependant plus de temps à faire le trajet qui la mène au travail et elle peut arriver en retard. Une grève des trains peut également entraîner des embouteillages et la retarder. La probabilité que la réunion commence à 9h dépend donc de la probabilité que John entende son réveil, de la probabilité qu'il y ait une grève des trains et de la probabilité qu'il y ait du verglas.

Les dépendances entre les variables de ce problème peuvent être représentées par le réseau bayésien décrit par la figure 6.2.

Supposons que la probabilité qu'il y ait une grève des trains soit de 5%, que le risque de verglas soit de 10% et que John n'entende son réveil que 3 fois sur 5. De plus, s'il y a du verglas, celui-ci retarde Marie 7 fois sur 10. Lorsqu'il y a des grèves, il y a 30% de chances qu'elles s'accompagnent de bouchons retardant Marie. Enfin, si John n'entend pas son réveil, il est en retard 7 fois sur 10 et en cas de grève John est 9 fois sur 10 en retard. Toutes ces informations quantitatives peuvent être représentées par des distributions de probabilités conditionnelles comme illustré sur la figure 6.3.

[34] Les diagrammes d'influence proposent une extension des réseaux bayésiens plus adaptée à la résolution de certains problèmes de décision car ils permettent la représentation des actions et des utilités.

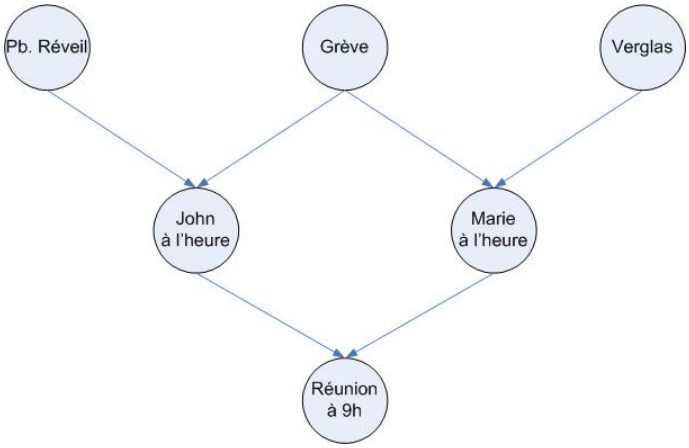


FIG. 6.2 – Exemple de représentation par un réseau Bayésien

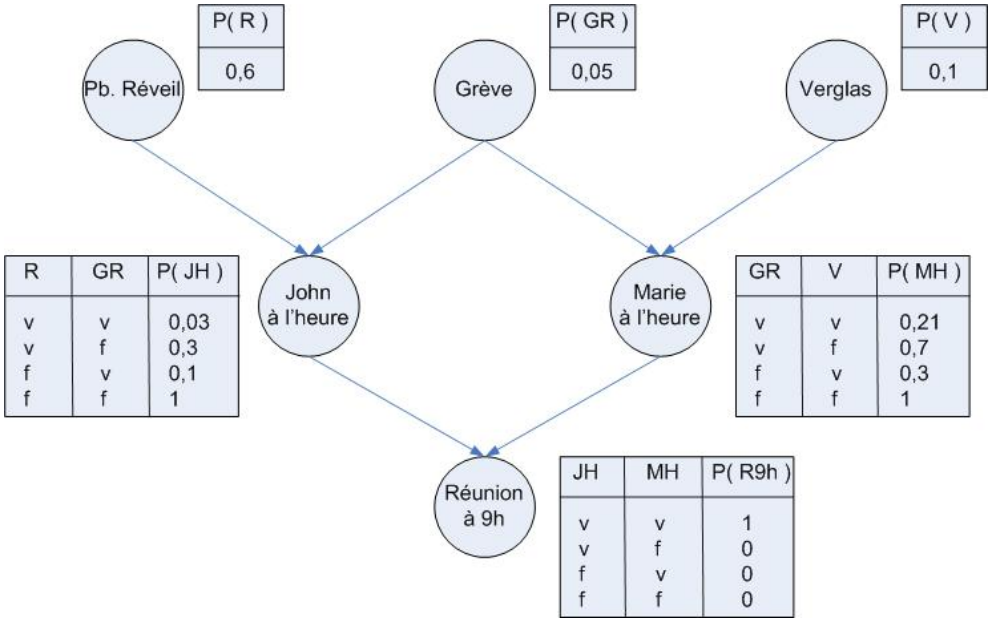


FIG. 6.3 – Exemple de représentation par un réseau Bayésien

A partir d'un tel réseau bayésien, nous pouvons inférer les probabilités de certains événements. Il est par exemple possible de prévoir le risque que la réunion ne commence pas à l'heure où que l'un des deux personnages de notre scénario soit en retard. Pour plus de détails concernant les réseaux bayésiens, le lecteur peut se référer à [Pearl, 1981].

6.2.2 Probabilités de transition et réseaux Bayésiens

Revenons à présent au calcul des probabilités sur les dates de début et de fin d'exécution des tâches. Les dépendances entre les variables sur lesquelles portent ces probabilités (dates de début et dates de fin) peuvent aisément être représentées par un réseau bayésien. En effet, la date de fin d'exécution d'une tâche t_{i+1} dépend de sa date de début et de la durée d'exécution de la tâche. La date de début de la tâche t_{i+1} est conditionnée par les dates de fin de ses prédécesseurs et par la politique de l'agent $\mathcal{A}g_i$ chargé de l'exécuter. Cette dernière dépend quant à elle de la date de fin de la précédente tâche t_i exécutée par l'agent $\mathcal{A}g_i$, ainsi que des ressources disponibles.

A partir du graphe de la mission, nous pouvons construire un réseau bayésien représentant les relations entre ces variables. Il suffit pour cela de remplacer chaque nœud du graphe par la règle de construction décrite par la figure 6.4 et traduisant les dépendances qui viennent d'être citées.

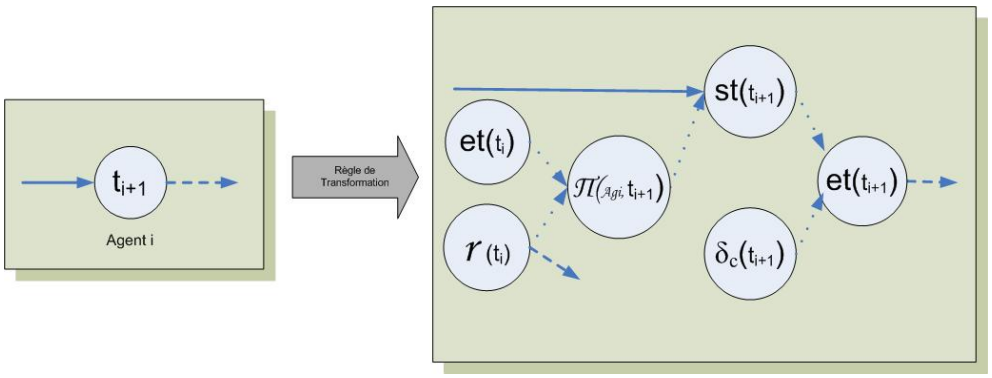


FIG. 6.4 – Règle de construction du réseau bayésien

En appliquant la règle précédente à chaque nœud du graphe d'une mission, nous obtenons un réseau bayésien décrivant les relations entre les différentes variables intervenant dans le calcul des fonctions de transition individuelles. Une telle transformation sur un graphe de mission complet est décrite par la figure 6.5.

Les tables de probabilités associées aux nœuds du réseau bayésien recensent, pour chaque variable, l'ensemble des valeurs qu'elle peut prendre. Par exemple, il sera associé à un nœud $st(t_{i+1})$, l'ensemble des dates de début possibles de la tâche t_{i+1}. A chacune de ces dates st correspondra une probabilité : la probabilité que la tâche t_{i+1} commence à st. De la même façon, à tout nœud $et(t_{i+1})$ est associée une table fournissant, pour chaque date de fin et de t_{i+1}, la probabilité que la tâche finisse à et.

Les probabilités associées aux nœuds du réseau bayésien représentant des dates de début ou des dates de fin correspondent donc aux probabilités que nous cherchons à calculer. Établir ces probabilités revient par conséquent à déterminer les tables de probabilités associées à chaque

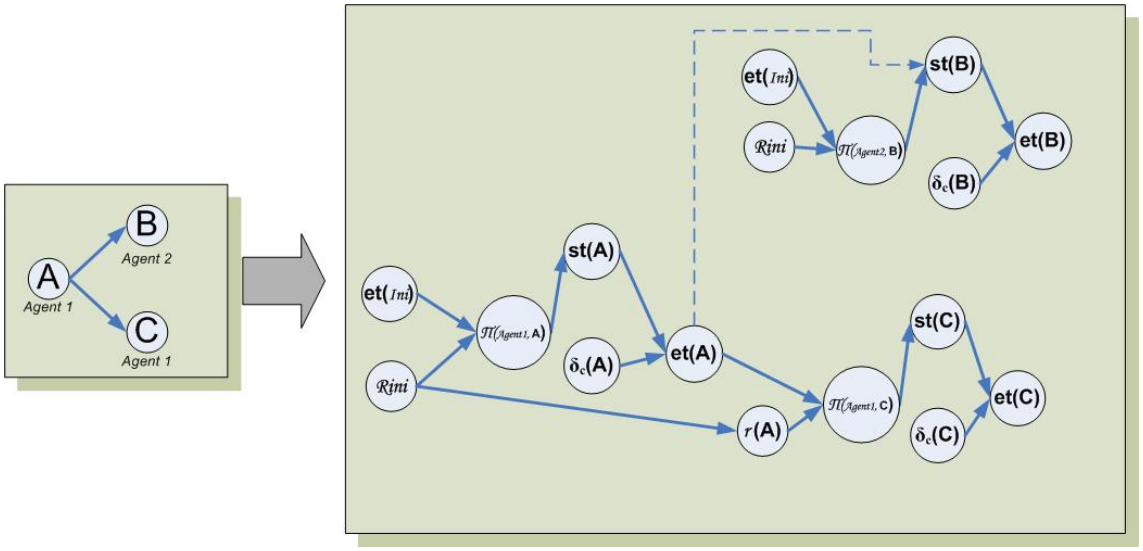


FIG. 6.5 – Transformation d'un graphe de mission en réseau bayésien

nœud du réseau.

6.2.3 Probabilités conditionnelles et simples

Soit t_{i+1} une tâche exécutée par un agent $\mathcal{A}g_i$ et t_i la tâche exécutée par $\mathcal{A}g_i$ juste avant t_{i+1}. Comme le décrit la règle de transformation 6.4, la date de fin d'une tâche t_i ainsi que les ressources disponibles influencent la date de début de la tâche suivante t_{i+1}. De ce fait, nous définirons les probabilités sur les dates de début de t_{i+1} comme des probabilités conditionnelles dépendant de la date de fin $et(I')_{t_i}$ de t_i et de la quantité r de ressources disponibles.

Définition 23 $P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r)$ *désigne la probabilité que l'exécution de la tâche* t_{i+1} *commence à* st *sachant que l'exécution de la tâche* t_i *s'est achevée à* $et(I')_{t_i}$ *et qu'il reste* r *ressources à l'agent.*

A partir de ces probabilités, nous pourrons également définir les probabilités sur les dates de début en supposant uniquement connue la date de fin de la tâche précédente.

Définition 24 $P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i})$ *désigne la probabilité que l'exécution de la tâche* t_{i+1} *commence à* st *sachant que l'exécution de la tâche* t_i *s'est achevée à* $et(I')_{t_i}$.

Les probabilités conditionnelles seront opposées aux probabilités simples $P_{\mathcal{ST}}^{t_{i+1}}(st)$ ne supposant aucune connaissance de la date de fin de la dernière tâche t_i, ni de la quantité de ressources disponibles r.

Définition 25 $P_{\mathcal{ST}}^{t_{i+1}}(st)$ *désigne la probabilité que l'exécution de la tâche* t_{i+1} *commence à* st.

De la même façon, nous définirons un ensemble de probabilités conditionnelles et simples sur les dates de fin d'exécution des tâches. La date de fin de t_{i+1} dépend de sa date de début et de sa durée d'exécution. La date de début t_{i+1} étant dépendante de la date de fin de t_i et des ressources disponibles, il est possible de définir les probabilités sur les dates de fin d'exécution de t_{i+1} en fonction de la date de fin de t_i et de la quantité de ressources disponibles.

Définition 26 $P_{\mathcal{ET}}^{t_{i+1}}(et|et(I')_{t_i}, r)$ *désigne la probabilité que l'exécution de la tâche* t_{i+1} *s'achève à et sachant que l'exécution de la tâche* t_i *s'est achevée à* $et(I')_{t_i}$ *et qu'il reste* r *ressources à l'agent.*

Définition 27 $P_{\mathcal{ET}}^{t_{i+1}}(et|et(I')_{t_i})$ *désigne la probabilité que l'exécution de la tâche* t_{i+1} *s'achève à et sachant que l'exécution de la tâche* t_i *s'est achevée à* $et(I')_{t_i}$ *et qu'il reste* r *ressources à l'agent.*

Définition 28 $P_{\mathcal{ET}}^{t_{i+1}}(et)$ *désigne la probabilité que l'exécution de la tâche* t_{i+1} *s'achève à et.*

Dans les sections qui suivent, nous allons plus particulièrement nous intéresser au calcul de ces probabilités sur les dates de début et de fin d'exécution des tâches [Beynier et Mouaddib, 2005a, Beynier et Mouaddib, 2004c] que nous désignerons par le terme de probabilités temporelles.

6.3 Calcul des probabilités temporelles

Lors de la construction du précédent réseau bayésien, le recours à des variables représentant la politique d'exécution de chaque tâche a été nécessaire. Il a ainsi été possible de formaliser l'influence des politiques sur les dates de début des tâches. A partir des politiques des agents qui ont été initialement fixées par notre hypothèse de décomposition de la fonction de transition, nous allons déterminer les probabilités associées aux nœuds $\pi_i(\mathcal{A}g_i, t_{i+1})$ du réseau bayésien.

6.3.1 Représentation probabiliste des politiques des agents

Pour chaque date de début st d'une tâche t_{i+1}, la table associée à une variable $\pi_i(\mathcal{A}g_i, t_{i+1})$ nous donne la probabilité que la politique de l'agent $\mathcal{A}g_i$ lui dicte de commencer l'exécution de t_{i+1} à st. Cette probabilité dépend des valeurs des parents du nœud, c'est-à-dire de la date de fin de la précédente tâche t_i exécutée par $\mathcal{A}g_i$ et des ressources disponibles.

L'influence de ces variables sur la politique a déjà été formulée dans la partie précédente de ce document. En effet, nous avons défini la politique d'un agent comme une fonction associant à chaque état de l'agent, une action. Un état étant défini comme l'ensemble [tâche précédente, [date de début, date de fin], ressources], nous retrouvons l'influence, sur la politique, de la date de fin de la dernière tâche exécutée par l'agent (tâche précédente) et des ressources disponibles.

La politique d'un agent étant définie comme une fonction de l'espace d'états vers l'espace d'actions, construire les tables de probabilités associées aux variables politiques revient à établir une représentation probabiliste de cette fonction. Ainsi, la probabilité $P_{\pi_i}(st|et(I')_{t_i}, r_{t_i})$ que la politique π_i de l'agent $\mathcal{A}g_i$ lui dicte de commencer t_{i+1} à st, lorsque t_i s'est achevée à $et(I')$ avec r_{t_i} ressources, est déduite directement à partir de l'ensemble des politiques initiales, en utilisant les règles suivantes :

- Si $\pi_i([t_i, [*, et(I')], r_{t_i}]) = st$ alors $P_{\pi_i}(st|et(I')_{t_i}, r_{t_i}) = 1$ où π_i désigne la politique de l'agent $\mathcal{A}g_i$.
- Sinon $P_{\pi_i}(st|et(I')_{t_i}, r_{t_i}) = 0$

Les politiques considérées étant déterministes, la représentation probabiliste ne comprendra que des 0 et des 1.

6.3.2 Probabilités sur les dates de début

Une fois la représentation probabiliste des politiques établie, nous pouvons déterminer les probabilités associées aux dates de début et de fin d'exécution des tâches. Nous supposerons pour le moment que les probabilités sur les ressources disponibles sont connues.

La probabilité que l'agent $\mathcal{A}g_i$ commence la prochaine tâche t_{i+1} à st dépend de la date à laquelle les prédécesseurs de t_{i+1} terminent leur exécution, ainsi que de l'état dans lequel se trouve l'agent. Afin d'entreprendre l'exécution de la tâche t_{i+1}, l'agent $\mathcal{A}g_i$ doit se trouver dans un état de succès ou dans un état d'échec partiel. Supposons que t_i soit la dernière tâche réalisée par un agent $\mathcal{A}g_i$. Lorsque $\mathcal{A}g_i$ commence l'exécution de la tâche suivante t_{i+1}, soit il est dans un état d'échec partiel $s_i = [t_i, [*, *], et(I')_{t_i}, r]$, soit il est dans un état de succès $s_i = [t_i, [, *, et(I')_{t_i}], r]$. La date de début de t_{i+1} va donc dépendre de la politique en s_i.

L'agent $\mathcal{A}g_i$ exécutera la tâche t_{i+1} à st si :

- sa politique lui dicte de commencer t_{i+1} à st et les prédécesseurs de t_{i+1} ont terminé leur exécution,
- sa politique lui dicte de commencer t_{i+1} à st' ($st' < st$), l'agent échoue partiellement (les prédécesseurs n'ont pas terminé) et après 0 à N échecs partiels sa politique lui dicte de commencer l'exécution de t_{i+1} à st.

De ce fait, $P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r)$ est définie de la manière suivante :

$$
\begin{aligned}
P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r) = & P_{enough}^{r}(Pred(t_{i+1})). \prod_{a \in P_{red}(t_{i+1}) \setminus t_i} P_{\mathcal{ET}}^{a}(\delta_e \leq st(I)|et(I')_{t_i}) \\
& \cdot \Big(P_{\pi_i}(st|t_i, et(I')_{t_i}, r) + \sum_{et(I') \leq st' < st}^{st} P_{\pi_i}(st'|t_i, et(I')_{t_i}, r) \\
& \cdot P_{not\_end}(st') \cdot P_{\mathcal{ST}}^{s_i}(st, s_i = [t_i, [st, st+1], et(I')_{t_i}, r - \Delta r']) \Big) \qquad (6.1)
\end{aligned}
$$

où $P_{not\_end}(st')$ désigne la probabilité que les prédécesseurs de la tâche t_{i+1} n'aient pas terminé à st'. Nous reviendrons par la suite sur le calcul de cette probabilité.

Pour que la tâche puisse commencer il est nécessaire que les prédécesseurs aient terminé à st. La probabilité d'être dans un tel cas est définie par :

$$P^r_{enough}(Pred(t_{i+1})) \cdot \prod_{a \in P_{red}(t_{i+1})} P^a_{\mathcal{ET}}(\delta_e \leq st(I) | et(I')_{t_i})$$

où $P^r_{enough}(Pred(t_{i+1}))$ désigne la probabilité que les prédécesseurs aient assez de ressources.

Comme le décrit l'équation 6.1, nous envisageons les cas où la politique de l'agent lui dicte de commencer t_{i+1} à st $(P_{\pi_i}(st|t_i, et(I'), r))$. Puis, nous considérons les cas où la politique de l'agent lui dicte de commencer avant st, l'agent échoue partiellement et après un ou plusieurs échecs partiels il commence l'exécution de la tâche à st.

$\Delta r'$ désigne la consommation de ressources d'un échec partiel. Afin de ne pas surcharger les formules nous supposerons que cette consommation est déterministe et identique pour toutes les tâches. Nos propos peuvent cependant être étendus au cas où les consommations des échecs partiels sont stochastiques (de la même façon que les consommations de ressources dues à l'exécution des tâches).

$P^{s_i}_{\mathcal{ST}}(st, s_i)$ correspond à la probabilité qu'à partir d'un état $s_i = [t_i, [st, st+1], et(I')_{t_i}, r_{t_i} - \Delta r']$ la politique de l'agent lui dicte de commencer à st. Entre temps l'agent peut passer par 0 à N échecs partiels. Nous obtenons ainsi :

$$\begin{aligned} P^{s_i}_{\mathcal{ST}}(st(I), s_i) = P_{\pi_i}(st|t_i, st+1, et(I')_{t_i}, r_{t_i} - \Delta r') + \\ \sum_{st'=st+1}^{st} P_{\pi_i}(st'|t_i, [st, st+1], et(I')_{t_i}, r_{t_i} - \Delta r') \cdot P_{not\_end}(st') \\ \cdot P^{s_i}_{\mathcal{ST}}(st, s_i = [t_i, [st', st'+1], et(I')_{t_i}, r_{t_i} - \Delta r' - \Delta r'])\Big) \end{aligned}$$

Enfin, la probabilité $P^r_{enough}(Pred(t_{i+1}))$ est telle que :

$$P^r_{enough}(Pred(t_{i+1})) = \prod_{t_k \in Pred(t_{i+1}) \setminus t_i} P^r_{enough}(t_k)$$

où $P^r_{enough}(t_k)$ désigne la probabilité que l'agent $\mathcal{A}g_k$ ait assez de ressources pour exécuter la tâche t_k. Cette probabilité est définie de la façon suivante :

– Si t_k est la dernière tâche que l'agent $\mathcal{A}g_k$ doit exécuter :

$$P^r_{enough}(t_k) = \sum_{r_{t_k} \geq 0} \sum_{\Delta^k_r | r_{t_k} - \Delta r \geq 0} P^{t_k}_{ra}(r_{t_k}) \cdot P_r(\Delta^k_r)$$

où $P_{ra}^{t_k}(r_{t_k})$ est la probabilité que l'agent $\mathcal{A}g_k$ ait r_{t_k} ressources lorsqu'il entreprend l'exécution de t_k. Le calcul de cette probabilité sera détaillé dans la section suivante.

– Sinon :

$$P_{enough}^{r}(t_k) = \sum_{r_{t_{k+1}} \geq 0} P_{ra}^{t_{k+1}}(r_{t_{k+1}})$$

où t_{k+1} est la prochaine tâche de l'agent $\mathcal{A}g_k$.

A partir des probabilités conditionnelles $P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r)$, nous pouvons déduire les valeurs des probabilités conditionnelles $P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i})$ et des probabilités simples $P_{\mathcal{ST}}^{t_{i+1}}(st)$. Ainsi :

$$P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}) = \sum_{r_{t_i}} P_{ra}^{t_i}(r_{t_i}).P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r_{t_i}) \tag{6.2}$$

Et,

$$P_{\mathcal{ST}}^{t_{i+1}}(st) = \sum_{et(I')_{t_i} \in ET(t_i)} P_{\mathcal{ET}}^{t_i}(et(I')_{t_i}) \sum_{r_{t_i}} P_{ra}^{t_i}(r_{t_i}).P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r_{t_i}) \tag{6.3}$$

6.3.3 Probabilités sur les dates de fin

Une fois les probabilités sur les dates de début d'une tâche t_{i+1} calculées, nous sommes en mesure de déterminer les probabilités $P_{\mathcal{ET}}^{t_{i+1}}$ sur les dates de fin de la tâche.

Probabilités simples

La probabilité que l'exécution de la tâche t_{i+1} termine à et est égale à la somme des probabilités qu'elle commence à st et dure $\delta_c^{i+1} = et - st$ unités de temps :

$$P_{\mathcal{ET}}^{t_{i+1}}(et) = \sum_{st \in ST(t_i)} \sum_{\delta_c^{i+1} | st + \delta_c^{i+1} = et} P_{\mathcal{ST}}^{t_{i+1}}(st) \cdot P_c(\delta_c^{i+1}) \tag{6.4}$$

Probabilités conditionnelles

De la même façon, la probabilité que la tâche termine à et sachant que t_i s'est terminée à $et(I')_{t_i}$ s'obtient à partir des probabilités conditionnelles sur les dates de début st et des probabilités sur les durées d'exécution δ_c^{i+1} de t_{i+1}. Nous obtenons alors :

$$P_{\mathcal{ET}}^{t_{i+1}}(et|et(I')_{t_i}, r) = \sum_{st \in ST(t_i)} \sum_{\delta_c^{i+1} | st + \delta_c^{i+1} = et} P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}, r) \cdot P_c(\delta_c^{i+1}) \tag{6.5}$$

Et,

$$P_{\mathcal{ET}}^{t_{i+1}}(et|et(I')_{t_i}) = \sum_{st \in ST(t_i)} \sum_{\delta_c^{i+1} | st + \delta_c^{i+1} = et} P_{\mathcal{ST}}^{t_{i+1}}(st|et(I')_{t_i}) \cdot P_c(\delta_c^{i+1}) \tag{6.6}$$

6.4 Propagation des probabilités sur les ressources

Intéressons nous à présent au calcul des probabilités portant sur les ressources disponibles, considérées jusqu'à présent comme connues.

Rappelons tout d'abord que les ressources envisagées ne sont pas partagées entre les agents. Chaque agent possède initialement une quantité R_{ini} de ressources[35] qui se trouve consommée au fil de l'exécution des tâches et des échecs partiels. La quantité de ressources disponible par $\mathcal{A}g_i$ à la fin de l'exécution de t_i correspond à la quantité de ressources disponible par l'agent avant toute tentative d'exécution de la tâche t_{i+1} (t_i étant la tâche exécutée par $\mathcal{A}g_i$ avant t_{i+1}). Après l'exécution de t_{i+1}, cette quantité de ressources est amputée des consommations des échecs partiels de t_{i+1} et de la consommation de l'exécution de t_{i+1}.

Soit $P_{ra}^{t_{i+1}}(r_{t_{i+1}})$ la probabilité que l'agent $\mathcal{A}g_i$ exécutant t_{i+1} dispose de $r_{t_{i+1}}$ ressources avant toute tentative d'exécution de t_{i+1}. Cette probabilité correspond donc à la probabilité qu'il reste à l'agent $\mathcal{A}g_i$ une quantité $r_{t_{i+1}}$ de ressources après l'exécution de t_i. Les probabilités P_{ra} peuvent alors être définies à l'aide des équations suivantes :

1. **Si t_{i+1} est la première tâche de l'agent ($t_i = \emptyset$) :**

$$P_{ra}^{t_{i+1}}(r_{t_{i+1}}) = \begin{cases} 0 & \text{si } r_{t_{i+1}} \neq R_{ini} \\ 1 & \text{sinon} \end{cases}$$

   En effet, avant toute tentative d'exécution de la première tâche d'un agent, aucune ressource n'a été consommée. Il reste donc à l'agent une quantité R_{ini} de ressources.

2. **Si t_{i+1} est la deuxième tâche de l'agent ($t_{i-1} = \emptyset$ et $t_i \neq \emptyset$)** (t_{i-1} désigne la tâche exécutée par $\mathcal{A}g_i$ avant t_i) :

$$P_{ra}^{t_{i+1}}(r_{t_{i+1}}) = \overbrace{\sum_{r_{t_i}} P_{ra}^{t_i}(r_{t_i})}^{\text{ressources disponibles avant l'exécution}} \cdot \overbrace{\sum_{nbEP} P_{EP}(nbEP, r_{t_i})}^{\text{nb. d'échecs partiels}} \cdot \overbrace{\sum_{\Delta_r^i | r_{t_i} - nbEP \cdot \Delta r' - \Delta_r^i = r_{t_{i+1}}} P_r(\Delta_r^i)}^{\text{consommation de ressources}}$$

   Afin de déterminer les probabilités sur les ressources disponibles avant toute tentative d'exécution de t_{i+1}, nous devons calculer les probabilités sur les ressources disponibles après l'exécution de t_i. Celles-ci sont déterminées à partir des ressources disponibles avant l'exécution de la tâche, en tenant compte du nombre d'échecs partiels, de leur consommation de ressources $\Delta r'$ et de la consommation de l'exécution de t_i.

[35] Nous supposons ici que tous les agents disposent initialement d'une même quantité de ressources. Nous pouvons toutefois gérer des situations où les quantités de ressources initialement disponibles varient d'un agent à l'autre.

Pour chaque taux de ressources disponibles avant l'exécution de t_{i+1}, nous envisageons toutes les possibilités d'échec partiel. $P_{EP}(nbEP, r_{t_i})$ est donc égale à la probabilité qu'il y ait eu $nbEP$ échecs partiels avant l'exécution de t_i. En considérant l'ensemble des consommations des échecs partiels et des durées d'exécution de t_i, nous pouvons alors déterminer les probabilités sur les quantités de ressources disponibles après l'exécution de t_i et donc avant toute tentative d'exécution de t_{i+1}. Lorsque l'exécution de la tâche réussit dès la première tentative alors, $nbEP = 0$. Les ressources disponibles après l'exécution de t_i sont donc obtenues en retranchant, aux ressources disponibles avant l'exécution de t_i, les consommations possibles de t_i.

3. **Sinon** :

$$P_{ra}^{t_{i+1}}(r_{t_{i+1}}) = \overbrace{\sum_{r_{t_i}} P_{ra}^{t_i}(r_{t_i})}^{\text{ressources disponibles avant l'exécution}} \cdot \overbrace{\sum_{et_{i-1}} P_{\mathcal{ET}}^{t_{i-1}}(et_{i-1}) \sum_{nbEP} P_{EP}(nbEP|et_{i-1}, r_{t_i})}^{\text{nb. d'échecs partiels}} \cdot \overbrace{\sum_{\Delta r_{t_i} | r_{t_i - nbEP \cdot \Delta r' - \Delta_r^i} = r_{t_{i+1}}} P_r(\Delta_r^i)}^{\text{consommation de ressources}}$$

Cette équation est quasiment identique à l'équation précédente. Néanmoins, l'agent $\mathcal{A}g_i$ a exécuté une tâche t_{i-1} avant t_i, nous tenons donc compte de l'influence de la date de fin de t_{i-1} sur la date de début de t_i et sur le nombre d'échecs partiels ayant eu lieu.

$P_{EP}(nbEP|r_{t_i})$ désigne la probabilité que la tâche s'exécute après nbEP échecs partiels lorsque l'agent dispose de r_{t_i} ressources. Nous considérons toutes les dates de début possibles de t_i avant lesquelles nbEP échecs partiels peuvent avoir eu lieu. Cette probabilité est définie de la façon suivante :

$$P_{EP}(nbEP, r_{t_i}) = \frac{\sum_{st_i \in ST(t_i) | nbEP \in EP(st_i)} P_{\mathcal{ST}}^{t_i}(st_i|r_{t_i}) \cdot \sum_{\delta_c^{i+1} | st_i + \delta_c^{i+1} \leq LET_{i+1}} P_c(\delta_c^{i+1})}{(1 - \sum_{st_i \in ST(t_i)} \cdot P_{\mathcal{ST}}^{t_i}(st_i)) \cdot (1 - \sum_{\delta_c^{i+1} | st_i + \delta_c^{i+1} \leq LET} P_c(\delta_c^{i+1}))}$$

où $EP(st_i)$ retourne les nombres d'échecs partiels pouvant avoir eu lieu avant que la tâche t_i commence son exécution à st_i. Cette fonction utilise la politique de la tâche t_i afin de déterminer les plans possibles pouvant mener à une exécution de t_i à st_i. Elle procède alors, pour chaque plan, à un dénombrement du nombre d'échecs partiels. La probabilité P_{EP} est donc égale à la probabilité que la tâche commence à une de ces dates et qu'elle finisse avant la date de fin au plus tard, le tout normalisé par la probabilité que la tâche réussisse à s'exécuter (nous ne considérons que les cas où la tâche s'exécute).

De la même façon, nous avons :

$$P_{EP}(nbEP|et_{i-1}, r_{t_i}) = \frac{\sum_{st_i \in ST(t_i) | nbEP \in EP(st_i|et_{i-1})} P_{\mathcal{ST}}^{t_i}(st_i|et_{i-1}, r_{t_i}) \cdot \sum_{\delta_c^{i+1} | st_i + \delta_c^{i+1} \leq LET_{i+1}} P_c(\delta_c^{i+1})}{(1 - \sum_{st_i \in ST(t_i)} P_{\mathcal{ST}}^{t_i}(st_i|et_{i-1})) \cdot (1 - \sum_{\delta_c^{i+1} | st_i + \delta_c^{i+1} \leq LET_{i+1}} P_c(\delta_c^{i+1}))}$$

où $EP(st_i|et_{i-1})$ retourne les nombres d'échecs partiels pouvant avoir eu lieu avant que la tâche t_i commence son exécution à st_i, lorsque t_{i-1} s'est terminée à et_{i-1}.

6.5 Algorithme de propagation des probabilités

Les propos qui précèdent ont montré comment calculer les probabilités sur les dates de début et de fin d'exécution d'une tâche t_{i+1}, ainsi que les probabilités sur les ressources disponibles avant toute tentative d'exécution d'une tâche t_{i+1}. Nous allons à présent décrire un algorithme permettant de déterminer ces données pour l'ensemble des tâches de la mission.

Les formules qui viennent d'être présentées montrent que le calcul des probabilités pour une tâche t_{i+1} nécessite que les probabilités portant sur les prédécesseurs de t_{i+1} aient déjà été déterminées. Nous devons donc, comme dans le cadre du calcul des dates de début et de fin d'exécution (cf. algorithme 3), découper le graphe en niveaux et les parcourir l'un après l'autre en commençant par le niveau constitué des tâches racines du graphe. Pour chaque tâche t_{i+1}, le calcul des probabilités sur la disponibilité des ressources et des probabilités temporelles (lignes 5 à 14) est réalisé à l'aide des équations présentées dans les sections qui précèdent. L'algorithme ainsi obtenu est décrit par l'algorithme 5.

La complexité temporelle de cet algorithme est linéaire en espace d'états. En effet, l'algorithme parcourt l'ensemble des tâches de la mission. Pour chaque tâche t_{i+1}, nous considérons l'ensemble des taux de ressources disponibles avant l'exécution de la tâche ($\#r_{t_{i+1}}$ valeurs). Puis, chaque date de début de t_{i+1} et chaque date de fin est envisagée, soit une complexité totale de l'ordre de :

$$\sum_{t_{i+1}\in\mathcal{T}} |ST_{i+1}| \times |ET_i| \times \#r_{t_{i+1}} = |\mathcal{SUS}|$$

où $\mathcal{SUS}$ correspond à l'union des espaces d'états de succès des agents ($|\mathcal{SUS}| < \sum_{\mathcal{A}g_i\in\mathcal{A}g} |\mathcal{S}_i|$).

Nous pouvons déduire que la complexité temporelle de l'algorithme est en $O(|\mathcal{SUS}|)$.

En ce qui concerne la complexité spatiale, nous stockons pour chaque taux de ressources, chaque date de début et chaque date de fin, les probabilités conditionnelles et simples précédemment définies, soit un nombre de probabilités tel que :

$$\begin{aligned}
&\sum_{t_{i+1}\in\mathcal{T}} \Big(|ST_{i+1}| + |ST_{i+1}| \times |ET_i| \times \#r_{t_{i+1}} + |ST_{i+1}| \times |ET_i| + |ET_{i+1}| \\
&+ |ET_{i+1}| \times |ET_i| \times \#r_{t_{i+1}} \\
&+ |ET_{i+1}| \times |ET_i| \Big) \\
&\simeq 2|\mathcal{SUS}| + \sum_{t_i\in\mathcal{T}} \Big(|ST_{i+1}| + |ST_{i+1}| \times |ET_i| + |ET_{i+1}| + |ET_{i+1}| \times |ET_i| \Big)
\end{aligned}$$

Algorithme 5 : Algorithme de propagation des probabilités

Entrées : un problème $\mathcal{X}$ enrichi des dates de début, de fin, des intervalles d'exécution et des taux de ressources possibles pour chaque tâche

1 $Niv \leftarrow root$ `// tâches du niveau courant`
2 $NivSuiv \leftarrow \emptyset$ `// tâches à considérer au niveau suivant`
3 **répéter**
4 **pour tous les** $t_{i+1} \in Niv$ **faire**
5 **si** *les probabilités de tous les prédécesseurs de* t_{i+1} *ont été calculées* **alors**
 `// Calcul des données sur l'exécution de` t_{i+1}
6 **pour tous les** $r_{t_{i+1}}$ *de* t_{i+1} **faire**
7 Calculer les probabilités conditionnelles et absolues $P_{ra}^{t_{i+1}}$ associées à $r_{t_{i+1}}$ (cf. section 6.4)
8 **fin**
9 **pour tous les** $st \in ST(t_{i+1})$ **faire**
10 Calculer les probabilités conditionnelles et absolues $P_{\mathcal{ST}}^{t_{i+1}}$ associées à st (cf. équations 6.3, 6.1 et 6.2)
11 **fin**
12 **pour tous les** $et \in ET(t_{i+1})$ **faire**
13 Calculer les probabilités conditionnelles et absolues $P_{\mathcal{ET}}^{t_{i+1}}$ associées à et (cf. équations 6.4, 6.5 et 6.6)
14 **fin**
15 $NivSuiv \leftarrow NivSuiv \cup Succ(t_{i+1})$
16 **sinon**
 `// les prédécesseurs n'ont pas tous été considérés`
17 $NivSuiv \leftarrow NivSuiv \cup \{t_{i+1}\}$
18 **fin**
19 **fin**
20 $Niv \leftarrow NivSuiv$
21 $NivSuiv \leftarrow \emptyset$
22 **jusqu'à** $Niv = \emptyset$;

Sorties : $\mathcal{X}$ enrichi de l'ensemble des probabilités sur les dates de fin, de début et les ressources disponibles

Étant donné que $\sum_{t_{i+1} \in \mathcal{T}} |ST_{i+1}| \times |ET_i| << |\mathcal{SUS}|$ et $\sum_{t_{i+1} \in \mathcal{T}} |ET_{i+1}| \times |ET_i| << |\mathcal{SUS}|$, nous pouvons déduire que la complexité spatiale de l'algorithme est en $O(|\mathcal{SUS}|)$.

Notons que l'algorithme de calcul des probabilités ne nécessite pas la construction du réseau bayésien. En effet, les dépendances entre les données portant sur l'exécution des tâches sont formalisées par les équations permettant le calcul des probabilités. Le réseau bayésien ne fournit qu'une représentation de ces dépendances et sert de support à la construction et à la compréhension des équations.

Remarquons par ailleurs que les algorithmes 3 (calcul des dates de début et de fin d'exécution d'une tâche), 4 (calcul des disponibilités des ressources) et 5 (calcul des probabilités) parcourent le graphe de la mission de la même façon. Ils peuvent donc être fusionnés afin d'inférer, à partir d'un seul parcours du graphe, l'ensemble des données nécessaires à la construction des MDPs locaux. La trame de l'algorithme ainsi obtenu est décrit par l'algorithme 6.

Pour chaque tâche t_{i+1}, cet algorithme considère tous les taux de ressources disponibles avant l'exécution de t_{i+1} et toutes les consommations de ressources possibles de cette dernière. L'ensemble des dates de fin des prédécesseurs est alors envisagé afin de déterminer les dates de début de la tâche. Pour chaque date de début, nous considérons l'ensemble des durées d'exécution possibles et nous en déduisons les dates de fin de t_{i+1}. La complexité temporelle de l'algorithme est donc de l'ordre de :

$$\sum_{t_{i+1} \in \mathcal{T}} \#r_{t_{i+1}} \times \#conso_{i+1} \times |ET_i| \times \#duree_{i+1}$$

où $\#conso_{i+1}$ est égal au nombre de consommations possibles pour la tâche t_{i+1} et $\#duree_{i+1}$ désigne le nombre de durées possibles pour t_{i+1}. Dans le pire cas, tous les taux de ressources calculés sont supérieurs à zéro et toutes les dates de début et de fin envisagées respectent les contraintes temporelles, nous avons alors :

$$|\mathcal{SUS}| = \sum_{t_{i+1} \in \mathcal{T}} \#r_{t_{i+1}} \times \#conso_{i+1} \times |ET_i| \times \#duree_{i+1}$$

L'algorithme a par conséquent une complexité temporelle polynomiale en $|\mathcal{SUS}|$.

La complexité spatiale de cet algorithme est, quant à elle, du même ordre que celle de l'algorithme 5. En effet, la quantité de données à stocker est identique. La complexité spatiale de l'algorithme est donc en $O(|\mathcal{SUS}|)$.

Algorithme 6 : Algorithme d'inférence des données sur l'exécution des tâches

Entrées : un problème $\mathcal{X}$

1 $Niv \leftarrow root$ `// tâches du niveau courant`

2 $NivSuiv \leftarrow \emptyset$ `// tâches à considérer au niveau suivant`

3 **répéter**

4 **pour tous les** $t_{i+1} \in Niv$ **faire**

5 **si** *les probabilités de tous les prédécesseurs de* t_{i+1} *ont été calculées* **alors**

 `// Calcul des données portant associés à` t_{i+1}

6 Calcul des taux de ressources de t_{i+1} : à chaque fois qu'un nouveau taux de ressources $r_{t_{i+1}}$ est calculé, déterminer les probabilités conditionnelles et absolues $P_{ra}^{t_{i+1}}$ associées à $r_{t_{i+1}}$

7 Calcul des dates de début st et de fin et de t_{i+1} : à chaque fois qu'une nouvelle date de début ou une nouvelle date de fin est calculée, déterminer les probabilités conditionnelles et absolues qui y sont associées

8 $NivSuiv \leftarrow NivSuiv \cup Succ(t_{i+1})$

9 **sinon**

 `// les prédécesseurs n'ont pas tous été considérés`

10 $NivSuiv \leftarrow NivSuiv \cup \{t_{i+1}\}$

11 **fin**

12 **fin**

13 $Niv \leftarrow NivSuiv$

14 $NivSuiv \leftarrow \emptyset$

15 **jusqu'à** $Niv = \emptyset$;

Sorties : $\mathcal{X}$ enrichi des dates de début, de fin, des intervalles d'exécution et des taux de ressources possibles ainsi que de l'ensemble des probabilités sur ces données

6.6 Calcul de la fonction de transition

Une fois l'algorithme 5 exécuté nous connaissons, pour chaque tâche t_{i+1} de la mission, les probabilités sur les dates de début ($P_{\mathcal{ST}}^{t_{i+1}}$), sur les dates de fin ($P_{\mathcal{ET}}^{t_{i+1}}$) et sur la disponibilité des ressources (P_{ra}). Nous sommes donc en mesure, pour toute tâche t_{i+1} de déterminer la probabilité que ses prédécesseurs aient terminé leur exécution à st.

A partir de ces données, nous sommes désormais en mesure de définir la probabilité qu'un agent $\mathcal{A}g_i$ décide, à partir d'un état s_i, de réaliser l'action a_i consistant à exécuter la tâche t_{i+1} à st[36]. Supposons que l'exécution de la précédente tâche t_i de l'agent se soit terminée à $et(I')_{task_i}$ et qu'il reste à l'agent une quantité r de ressources. Nous allons alors calculer les probabilités que l'exécution de t_{i+1}, réussisse, échoue partiellement ou échoue totalement.

Afin de calculer ces probabilités, nous avons précédemment fixé les politiques des agents. La probabilité de passer d'un état s_i à un état s_i' en exécutant la tâche t_{i+1} à st doit en revanche être calculée indépendamment de toute politique pour la tâche t_{i+1}. En effet, nous ne souhaitons pas calculer la probabilité que l'agent passe d'un état s_i à un état s_i' en exécutant t_{i+1} à st lorsqu'il suit la politique π_i. Nous cherchons à déterminer la probabilité que l'agent passe d'un état s_i à un état s_i' lorsqu'il décide d'exécuter t_{i+1} à st. Dans le premier cas, la probabilité sur la date de début st de t_{i+1} est calculée, tout comme $P_{\mathcal{ST}}^{t_{i+1}}(st)$, en tenant compte de $\pi_i(\mathcal{A}g_i, t_{i+1})$. Dans le second cas, la probabilité de commencer à st dépend uniquement de la probabilités que les prédécesseurs de t_{i+1} aient terminé.

Nous avons précédemment tenu compte, lors du calcul de $P_{\mathcal{ST}}^{t_{i+1}}(st)$, des politiques des agents car nous souhaitions calculer les probabilités sur les dates de début et de fin d'exécution des tâches, sachant que chaque agent suivait une politique donnée. Ainsi, nous pouvions déduire les probabilités sur les dates de fin des prédécesseurs. Ici, nous ne supposons aucune politique pour t_{i+1}, nous calculons les probabilités de transition pour chaque date de début possible de t_{i+1}. De ce fait, nous n'utilisons pas $P_{\mathcal{ST}}^{t_{i+1}}(st)$ afin de déterminer la probabilité que la tâche t_{i+1} commence à st. Cette dernière sera tout simplement égale à la probabilité que les prédécesseurs aient terminé, soit :

$$P_{enough}^{r}(Pred(t_{i+1})). \prod_{a \in P_{red}(t_{i+1})} \sum_{t \in ET(a) | t \leq st} P_{\mathcal{ET}}^{a}(t | et(I')_{t_i})$$

6.6.1 Exécution réussie

Pour que l'agent $\mathcal{A}g_i$ réussisse l'exécution de t_{i+1} à st, plusieurs conditions doivent être satisfaites :

– l'agent doit pouvoir commencer l'exécution de la tâche à st (l'exécution des prédécesseurs de t_{i+1} doit être terminée),

[36] st est supposée être une date de début valide : $st \in ST(t_{i+1})$.

- l'agent doit avoir assez de ressources (la consommation Δ_r^{i+1} de l'exécution de la tâche doit être inférieure ou égale à r),
- les contraintes temporelles doivent être respectées : $st + \delta_c^{i+1} \leq LET_{i+1}$

La probabilité que la tâche soit exécutée avec succès peut alors être définie de la manière suivante :

$$P_{suc}(st|et(I')_{t_i}, r) = \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t|t \leq st} P_{\mathcal{ET}}^{t_k}(t|et(I)_{t_i}). \sum_{\Delta_r^{i+1}|r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1}|st+\delta_c^{i+1} \leq LET_{i+1}} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})$$

où $P_r(\Delta_r^{i+1})$ est la probabilité que l'exécution de la tâche consomme Δ_r^{i+1} ressources et $P_c(\delta_c^{i+1})$ est la probabilité que l'exécution de la tâche dure δ_c^{i+1} unités de temps.

6.6.2 Échec partiel

La probabilité d'échouer partiellement l'exécution d'une tâche est égale à la probabilité que les prédécesseurs n'aient pas terminé leur exécution et que l'agent ait assez de ressources pour se rendre compte de son échec partiel. En effet, si l'agent manque de ressources lors de l'échec partiel, il passe dans un état d'échec total.

Remarquons que si $st = UB_{i+1}$ et que les prédécesseurs n'ont pas terminé, nous ne sommes pas en présence d'un échec partiel mais d'un échec total. Il n'existe en effet pas de date de début supérieure à st, à laquelle l'agent pourra re-tenter l'exécution de la tâche sans violer les contraintes temporelles. Le cas où $st = UB_{i+1}$ n'est donc pas à considérer dans ce type de transition.

La probabilité $P_{not\_end}(st)$ que les prédécesseurs n'aient pas fini à st est égale à la probabilité qu'ils finissent après st ou ne finissent jamais en raison d'un échec total. Les prédécesseurs peuvent ne jamais terminer en raison d'un manque de ressources ou de la violation de leurs contraintes temporelles. La probabilité qu'un tel événement arrive est définie par :

$$1 - P_{enough}^r(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{end \in ET(t_k)|et \leq UB_{i+1}} P_{\mathcal{ET}}^{t_k}(end|et(I')_{t_i})$$

La probabilité que les prédécesseurs finissent après st est, quant à elle, calculée de la manière suivante :

$$P_{enough}^r(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{end \in ET(t_k)} P_{\mathcal{ET}}^{t_k}(end|et(I')_{t_i})$$

$$-P_{enough}^r(Pred(t_{i+1})). \prod_{a \in P_{red}(t_{i+1}) \setminus t_i} \sum_{t \in ET(a)|t \leq st} P_{\mathcal{ET}}^{a}(t|et(I')_{t_i})$$

$P_{not\_end}(st)$ est donc définie comme suit :

$$\begin{aligned} P_{not\_end}(st) =& P^r_{enough}(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{end \in ET(t_k) | et \leq UB_{i+1}} P^{t_k}_{\mathcal{ET}}(end | et(I')_{t_i}) \\ & - P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{\mathcal{ET}}(t | et(I')_{t_i}) \\ & + 1 - P^r_{enough}(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{end \in ET(t_k) | et \leq UB_{i+1}} P^{t_k}_{\mathcal{ET}}(end | et(I')_{t_i}) \end{aligned}$$

A partir des probabilités P_{not_end}, nous pouvons déduire la probabilité P_{PCV} d'une transition vers un échec partiel :

- Si $st < UB_{i+1}$:

$$P_{PCV} = \sum_{\Delta'_r | r \geq \Delta'_r} P_r(\Delta'_r) \cdot P_{not\_end}(st)$$

- Sinon :

$$P_{PCV} = 0$$

6.6.3 Échec total

Il existe deux raisons pour lesquelles l'exécution de la tâche t_{i+1} peut échouer totalement : un manque de ressources ou bien la violation des contraintes temporelles. Ces deux causes d'échec doivent donc être prises en compte lors du calcul de la probabilité de transition vers un état d'échec total.

• **Manque de ressources (MR) :**

Un agent peut manquer de ressources lors de l'exécution d'une tâche où lors d'un échec partiel. La probabilité qu'il manque de ressources lors de l'exécution d'une tâche est égale à la probabilité que la tâche puisse commencer mais que la consommation de son exécution soit supérieure aux ressources disponibles. La probabilité de manquer de ressources lors d'un échec partiel est égale à la probabilité que les prédécesseurs n'aient pas terminé leur exécution et que l'agent n'ait pas assez de ressources pour s'apercevoir qu'il ne peut pas exécuter la tâche (la consommation de l'échec partiel est supérieure aux ressources disponibles). Ainsi, nous obtenons :

- Si $st < UB_{i+1}$:

$$\begin{aligned} P_{MR} = P^r_{enough}(Pred(t_{i+1})) \cdot & \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{\mathcal{ET}}(t | et(I)_{t_i}) \cdot \sum_{\Delta^{i+1}_r | r < \Delta^{i+1}_r} P_r(\Delta^{i+1}_r) \\ & + P_{not\_end}(st). \sum_{\Delta'_r | r < \Delta'_r} P_r(\Delta'_r) \end{aligned}$$

où $\Delta r'$ correspond à la consommation de ressources d'un échec partiel et Δ^{i+1}_r est la consommation de l'exécution de la tâche t_{i+1}.

– Sinon ($st = UB_{i+1}$) :

$$P_{MR} = P^r_{enough}(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1}) \setminus t_i} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{\mathcal{ET}}(t|et(I)_{t_i}) \cdot \sum_{\Delta^{i+1}_r | r < \Delta^{i+1}_r} P_r(\Delta^{i+1}_r)$$

L'agent ne peut échouer partiellement, tout échec est total puisqu'il n'existe pas d'autres dates de début pour la tâche.

• **Contraintes temporelles $TC(t_{i+1})$ non respectées :**

Les contraintes temporelles imposent des restrictions sur les dates de début et de fin possibles de chaque tâche. Il existe deux façons de violer ces contraintes :

– soit la tâche commence trop tard

– soit la tâche finit trop tard

Les dates de début st d'une tâche t_{i+1} sont telles que $st \in ST(t_{i+1})$, c'est-à-dire $st \in [LB_{i+1}, UB_{i+1}]$. Une tâche va donc commencer trop tard (TT) si $st = UB_{i+1}$ et que l'exécution échoue car les prédécesseurs n'ont pas terminé. En effet, dans ce cas, toute date de début ultérieure à st ne respectera pas les contraintes temporelles. L'agent ne pourra alors pas re-tenter l'exécution de sa tâche sans violer les contraintes temporelles, l'exécution de la tâche échoue totalement. La probabilité d'un tel échec est définie de la manière suivante :

– si $st < UB_{i+1}$: $P_{TT} = 0$

– sinon : $P_{TT} = P_{not\_end}(st)$

Le second type de violation des contraintes temporelles concerne la violation de la date de fin au plus tard (LET). Dans ce cas, l'exécution de la tâche peut commencer, l'agent dispose d'assez de ressources mais la tâche se termine après LET_{i+1} : $st + \delta^{i+1}_c > LET_{i+1}$. La probabilité que l'agent arrive dans un état d'échec total pour cette raison est définie par P_{DM} (DM : Deadline Met) telle que :

$$P_{DM} = P^r_{enough}(Pred(t_{i+1})). \prod_{a \in P_{red}(t_{i+1}) - t_i} \sum_{t \in ET(a) | t \leq st} P^a_{\mathcal{ET}}(t|et(I)_{t_i}). \sum_{\Delta^{i+1}_r | r \geq \Delta^{i+1}_r} P_r(\Delta^{i+1}_r)$$

$$. \sum_{\delta^{i+1}_c | st + \delta^{i+1}_c > LET_{i+1}} P_c(\delta^{i+1}_c)_{i+1}$$

Pour finir, la probabilité que l'agent arrive dans un état d'échec total est égale à : $P_{TT} + P_{DM}$.

6.6.4 Complétude du système de transitions

La fonction de transition individuelle d'un agent $\mathcal{A}g_i$ définit les transitions possibles de l'agent $\mathcal{A}g_i$ lorsque celui-ci entreprend l'exécution d'une action a_i à partir d'un état s_i. Ce système de transition est complet c'est-à-dire qu'il prend en compte toutes les transitions possibles à partir d'un état s_i. Comme le montre la preuve du corollaire 1, la complétude du système peut être justifiée en réalisant la somme des probabilités des différents types de transition du système.

Corollaire 1 *Le système de transition est complet.*

La preuve du corollaire est détaillée dans l'annexe A.1 du document.

Conclusion

Dans ce chapitre, nous avons décrit le processus d'élaboration des fonctions de transition définies dans les MDPs composant un OC-DEC-MDP. A partir d'un problème à transitions dépendantes, nous avons réalisé une décomposition de la fonction de transition du système en fonctions de transition propres à chaque agent et indépendantes les unes des autres. Pour ce faire, nous avons fixé les politiques des agents et supposé qu'elles étaient connues de tous. Il a ainsi été possible de déterminer l'ensemble des probabilités nécessaires au calcul des fonctions de transition individuelles. A partir des probabilités (probabilités sur les dates de début, dates de fin et sur les ressources disponibles) ainsi inférées, nous avons été en mesure de déterminer pour chaque agent $\mathcal{A}g_i$, la probabilité qu'il passe d'un état s_i à un état s'_i en exécutant une action a_i (exécuter t_{i+1} à st). Une fonction de transition propre à chaque agent et indépendante des autres a alors pu être définie. Cette fonction suppose, pour chaque tâche t_{i+1}, que l'ensemble des tâches en amont de t_{i+1} sont exécutées en suivant les politiques des agents initialement fixées.

Dans la partie précédente de ce document, nous avons présenté les différentes composantes d'une modélisation sous forme d'OC-DEC-MDP. Dans ce chapitre, nous avons détaillé la construction des fonctions de transition nécessaires à la définition complète du modèle. Nous avons à présent à notre disposition toutes les informations nécessaires à la résolution du problème de planification des tâches que nous nous sommes posé. De ce fait, nous allons désormais pouvoir nous intéresser à la mise en place d'une méthode de résolution des OC-DEC-MDPs. Ce sujet fera l'objet des chapitres qui vont suivre.

Chapitre 7

Mesures pour l'évaluation des politiques

L'objectif premier de notre travail est de planifier l'exécution d'une mission de sorte que les agents maximisent leur espérance de gain collective. En raison des dépendances entre les tâches ainsi que des contraintes temporelles et de ressources, la maximisation de cette espérance de gain passe par une nécessaire coordination des politiques des agents. En effet, toute action d'un agent a des répercussions sur l'ensemble des membres du système. S'il n'en tient pas compte, l'agent agira de manière non-coopérative ce qui risque de conduire à un comportement sous-optimal du point de vue collectif.

Dans ce chapitre, nous allons nous intéresser à la prise de décision d'un agent. Afin de sélectionner l'action susceptible de maximiser l'espérance de gain collective, nous montrerons que celui-ci doit tenir compte de l'impact de ses choix sur lui même et sur les autres. Pour ce faire, nous définirons deux mesures. La première, basée sur le calcul de l'utilité espérée individuelle, permettra à l'agent de tenir compte du gain que l'exécution de ses actions peut espérer lui apporter. La seconde mesure, inspirée de la théorie économique du coût occasionné, permettra à l'agent d'agir de manière coopérative en tenant compte de l'impact de ses décisions sur les autres.

Nous procéderons tout d'abord à la formalisation de ces mesures puis, nous montrerons comment les conjuguer afin de décider quelle action exécuter à partir d'un état donné. Enfin, nous reviendrons plus en détail sur le calcul du coût occasionné et nous en discuterons différentes évaluations possibles.

7.1 Principe de l'évaluation des politiques

Résoudre un DEC-MDP de manière optimale consiste à déterminer la politique jointe qui maximise la mesure de performance des agents. Pour ce faire, il est nécessaire de parcourir entièrement l'espace des politiques jointes possibles et d'évaluer chacune d'entre elles. Bien que des méthodes de résolution aient été développées afin de diminuer en moyenne le nombre de politiques à évaluer, il demeure nécessaire d'envisager une grande partie des politiques.

Étant donnée la taille des problèmes que nous souhaitons résoudre, un tel parcours n'est pas réalisable, c'est pourquoi nous avons précédemment opté pour la recherche d'une solution approximant la solution optimale. Afin de traiter des problèmes de taille conséquente, nous avons choisi de décomposer tout problème $\mathcal{X}$ en un ensemble de problèmes plus simples à résoudre. Nous avons ainsi procéder à une modélisation des problèmes par un ensemble de MDPs, chaque MDP représentant le problème décisionnel d'un agent. Au lieu de résoudre un DEC-MDP de complexité NEXP, nous proposons ainsi de résoudre un ensemble de MDPs de complexité P-Complet.

7.1.1 Difficultés liées à la décomposition

La résolution de ces MDPs n'est toutefois pas sans poser de problèmes en ce qui concerne la coordination des politiques des agents. En effet, nous nous intéressons à des systèmes multi-agents coopératifs dans lesquels il existe de fortes dépendances entre les agents (dépendances liées aux contraintes de précédence, aux contraintes temporelles, etc.). Afin de maximiser au mieux les performances du système[37], il est primordial que les agents suivent des politiques coopératives.

Si les MDPs constituant notre modèle sont résolus indépendamment les uns des autres, les politiques obtenues ne tiendront pas compte de ces dépendances et génèreront un comportement non coordonné, le problème décisionnel de chaque agent étant résolu sans tenir compte des répercussions des décisions de l'agent sur les autres. Afin de maximiser au mieux les performances du système, il est donc nécessaire de mettre en place un mécanisme garantissant la coordination des politiques et la coopération entre les agents.

7.1.2 Mécanisme pour la prise en compte des influences entre les agents

L'obtention d'un comportement coopératif nécessite que chaque agent tienne compte de l'impact de ses décisions sur les autres agents. Supposons que vous deviez assister à une réunion à 10h et que celle-ci ne puisse débuter que si tous les participants sont présents. Afin que chaque participant perde le moins temps à attendre les autres, il est nécessaire que chacun adopte un comportement coopératif et prenne en compte l'impact de son retard sur les autres. Si tous vos collègues sont à l'heure et que vous arrivez avec un retard Δt, vous devez considérer le fait que

[37] Nous cherchons une solution approximant au mieux la solution optimale.

vous avez fait perdre Δt unités de temps aux autres participants. Leur perte en utilité (travail qu'ils auraient pu accomplir pendant ce temps) qui en découle rend un tel retard peu souhaitable. En revanche, si vous savez que vos collègues arriveront à 10h10, le temps que vous leur ferez perdre en arrivant avec 10 minutes de retard, est nul. Vous pouvez alors vous permettre un tel retard. Afin d'agir de manière coopérative, votre décision doit donc dépendre de la perte que vous générez sur vos collègues.

Il est possible d'envisager une solution pour laquelle la personne qui va être en retard de Δt unités de temps en avertit les autres. La date de début de la réunion pourra ainsi être repoussée et chacun pourra vaquer à ses occupations entre 10h et 10h+Δt. Néanmoins, cette solution n'est pas totalement satisfaisante. En effet, la perte provoquée sur les autres participants n'est pas seulement constituée de la perte de temps Δt, mais également des pertes sur les actions futures. Retarder la réunion à 10h+Δt peut par exemple empêcher certains collègues de participer à une autre réunion à 11h30. Avertir les autres de notre retard ne suffit donc pas, il est nécessaire de tenir compte des pertes futures provoquées par notre retard sur les autres membres du groupe.

Dans le cadre de notre travail, si un agent décide de retarder l'exécution d'une tâche t_{i+1}, il doit tenir compte du fait qu'il retardera également les successeurs de t_{i+1}. L'agent risque alors d'entraîner la violation des contraintes temporelles des tâche suivantes (augmentation de la probabilité que les successeurs de t_{i+1} finissent après leurs dates de fin au plus tard) ou de ne pas laisser assez de temps pour la réalisation des tâches restant à exécuter. La mesure de performance des agents s'en trouvera alors réduite.

Afin de coordonner les politiques d'un ensemble d'agents, Chadès et al. [Chadès *et al.*, 2002] se sont inspirés de la notion d'empathie. L'idée est de permettre à chaque agent de prévoir le comportement des autres agents et de s'y adapter en conséquence. Pour ce faire, les agents déterminent tour à tour leur politique, à partir des politiques des autres membres du système. Ces dernières sont supposées connues et sont utilisées afin de construire une modélisation du problème sous forme de MMDP ou de MDP Subjectif, ensuite résolu. Chaque fois qu'une nouvelle politique est calculée, une nouvelle modélisation du problème est réalisée et l'algorithme de résolution est à nouveau exécuté. Une méthode de résolution similaire a été décrite par Nair et al. [Nair *et al.*, 2003]. Dans un cadre décentralisé ces deux approches nécessitent que les agents communiquent leurs politiques chaque fois qu'elles sont modifiées. Une telle communication est très contraignante et limite les problèmes pouvant être traités.

Parmi les méthodes de coordination qui ont été proposées dans le domaine des MDPs, nous pouvons également citer l'approche de Dutech et al. [Dutech *et al.*, 2001] basée sur des techniques d'apprentissage. Elle permet un calcul simultané des politiques de tous les agents, tout en limitant les communications. Cependant, cette méthode requiert une phase d'apprentissage qui

n'est pas envisageable dans les applications auxquelles nous destinons notre travail.

Pour que chaque agent puisse tenir compte de l'impact de ses décisions sur les autres agents et qu'il adopte une politique coopérative, nous avons choisi d'avoir recours à une notion utilisée en sciences économiques : la notion de **coût occasionné**. Cette dernière nous a permis, comme nous allons le montrer par la suite, de mettre en place lors de la résolution des MDPs, un mécanisme de prise en compte de l'influence d'une décision sur les autres agents tout en limitant les communications entre les agents et les mises à jour de la modélisation du problème.

7.1.3 Coût occasionné et formalisation des interactions

La notion de coût occasionné est utilisée en théorie des choix économiques [Wieser, 1889] afin d'établir, sous contraintes de ressources, un choix entre différentes alternatives. Elle est exploitée afin de résoudre une grande diversité de questions économiques, comme par exemple pour établir les choix de production d'une entreprise, pour la gestion du temps, le management de carrières professionnelles ou encore afin de résoudre des questions d'investissement de capital.

Théorie du coût occasionné

Toute décision prise sous contraintes de ressources[38] (ressources limitées) nécessite qu'un choix soit réalisé entre les différentes possibilités qui s'offrent à nous. Celui-ci résulte alors d'un compromis entre le coût de l'action sélectionnée et son utilité.

La théorie du coût occasionné affirme que toute décision a un coût caché, et ignorer ce coût peut conduire à des choix erronés. En effet, le coût d'une action ne se traduit pas seulement en fonction des biens dépensés pour l'exécuter, mais également en fonction des bénéfices qui ne pourront être obtenus par la suite car cette action a été réalisée. Cette perte de bénéfices constitue le coût occasionné.

Par exemple, le coût lié à la décision de réaliser deux ans d'études dans une université étrangère prestigieuse n'est pas seulement constitué du coût de l'inscription et des différentes dépenses annexes (logement, livres, ...), il est également nécessaire de prendre en compte le coût occasionné de cette décision. Celui-ci correspond à la somme d'argent que vous auriez pu gagner en travaillant au lieu de réaliser des études. Il s'agit donc de la perte monétaire due au fait que vous ne pourrez pas travailler si vous choisissez de faire des études. Néanmoins, ce coût « réel » (coût des études + coût occasionné) est contrebalancé par la promesse d'un salaire plus important acquis grâce aux diplômes que vous aurez obtenus lors deux vos deux années d'études.

[38] Les ressources peuvent être de l'argent du temps, de l'énergie, ...

Votre décision résulte alors d'un compromis entre le coût « réel » et le gain de salaire que vous obtiendrez par la suite.

Le coût occasionné correspond ainsi à la perte provoquée par le fait que, une fois votre décision prise, une autre alternative ne pourra alors plus être choisie. Dans notre exemple, le coût occasionné s'exprime en terme d'argent. Plus généralement, il correspond à une perte d'utilité qui selon les biens dépensés peut également traduire une perte de temps, d'énergie, ...

Notre utilisation du coût occasionné

La théorie du coût occasionné a déjà été exploitée dans le cadre des processus décisionnels de Markov afin de planifier l'exécution d'unités de raisonnement progressif (PRU[39]) [Mouaddib et Zilberstein, 1998, Zilberstein et Mouaddib, 1999, Cardon *et al.*, 2001]. Ces travaux ont été appliqués avec succès à la recherche d'informations [Zilberstein *et al.*, 2000] et à la planification des actions d'un robot explorateur [Zilberstein et Mouaddib, 2000]. Dans le cadre d'un environnement dynamique, il est proposé de planifier l'exécution d'une PRU à l'aide d'un MDP en tenant compte des PRUs à venir via l'utilisation du coût occasionné. En effet, plus il est dépensé de ressources pour l'exécution de la PRU courante, moins il restera de ressources pour l'exécution des PRUs à venir. Le coût occasionné traduit l'impact de la diminution des ressources sur les PRUs restant à exécuter. Il correspond à la perte en utilité espérée due à cette diminution des ressources disponibles pour l'exécution des PRUs restantes. L'utilisation du coût occasionné permet alors la réalisation d'un compromis entre la quantité de ressources consommée pour exécuter la PRU courante et les répercussions de cette consommation sur les PRUs à venir.

Nous avons pour notre part fait appel à la notion de coût occasionné afin que chaque agent puisse prendre en compte le coût de ses décisions sur les autres agents. Tout agent dépense du temps pour exécuter une tâche t_{i+1}. Plus un agent retarde l'exécution d'une tâche t_{i+1}, plus il finira tard et plus les successeurs (directs et indirects) de t_{i+1} commenceront tard. Ce retard risque alors de provoquer une perte en utilité sur les autres agents que nous mesurons par le coût occasionné.

Reprenons l'exemple précédent concernant la participation à une réunion. Le coût occasionné va, dans ce cas, exprimer la perte en utilité provoquée sur un collègue lorsque vous arrivez avec un retard Δt. Cette perte en utilité traduit les bénéfices qui auraient pu être réalisés grâce à des alternatives devenues non-réalisables en raison de ce retard. Par exemple, si votre retard empêche un collègue d'assister à une autre réunion à 11h30, le coût occasionné traduira sa perte d'utilité induite par le fait qu'il ne peut participer à cette réunion.

Grâce à l'utilisation du coût occasionné chaque agent est en mesure de tenir compte des autres agents lorsqu'il doit décider quand exécuter une tâche. Toute décision résulte d'un compromis

[39] Progressive Processing Unit

entre l'utilité espérée de l'agent et le coût occasionné provoqué sur les autres agents, c'est-à-dire la perte en utilité suscitée par cette décision.

7.2 Evaluation d'une décision

Considérons un agent $\mathcal{A}g_i$ devant, à un instant t et à partir d'un état s_i, décider quelle action exécuter. Nous supposerons que la dernière tâche exécutée par l'agent est la tâche t_i, qu'elle s'est terminée à $et(I')_{t_i}$ et qu'il reste à l'agent une quantité r_{t_i} de ressources. L'état s_i est un état de succès ou un état d'échec partiel, donc $s_i = [t_i, [st(I')_{t_i}, et(I')_{t_i}], r_{t_i}]$ (nous avons alors $et(I')_{t_i} = t$) ou $s_i = [t_i, [t-1, t], et(I')_{t_i}, r_{t_i}]$.

Tout échec partiel consomme des ressources. Lorsque les ressources d'un agent sont limitées, celui-ci a tout intérêt à restreindre le nombre de ses échecs partiels. Pour ce faire, il peut choisir de retarder l'exécution de ses tâches afin d'augmenter la probabilité que leurs prédécesseurs aient terminé et diminuer la probabilité d'échouer partiellement. Retarder l'exécution d'une tâche t_{i+1} a cependant des répercussions à la fois sur l'agent $\mathcal{A}g_i$ et sur les autres agents. Ce retard entraîne tout d'abord une augmentation de la probabilité que l'agent $\mathcal{A}g_i$ finisse l'exécution de t_{i+1} après sa date de fin au plus tard. L'agent retarde de plus l'exécution de ses tâches suivantes, augmentant ainsi la probabilité de violer leurs contraintes temporelles (augmentation de la probabilité de commencer ou finir trop tard l'exécution des tâches suivantes de l'agent). Enfin, la décision de l'agent a pour conséquence de retarder les tâches accessibles à partir de t_{i+1} et exécutées par d'autres agents.

Reprenons le graphe de la figure 6.1. Si l'agent 2 décide de retarder l'exécution de la tâche B afin d'augmenter la probabilité que A soit terminée, il va alors également retarder l'exécution de ses prochaines tâches F et G, ainsi que l'exécution des tâches suivantes des autres agents (D pour l'agent 3 et E pour l'agent 1). Les autres agents (agent 1 et agent 3) sont donc influencés par la décision de l'agent 2. Ils risquent, en raison du retard de B, d'augmenter leur nombre d'échecs partiels et de ne pas pouvoir respecter les contraintes temporelles ou de ressources sur l'exécution de leurs tâches. La probabilité que l'exécution de leurs tâches échoue augmente alors et leur utilité espérée diminue. Lorsque l'agent $\mathcal{A}g_i$ décide de retarder l'exécution de t_{i+1}, il doit donc tenir compte de la diminution d'utilité provoquée sur les autres agents. Sa décision résulte alors d'un compromis entre les deux mesures que nous allons à présent détailler : son utilité espérée et la baisse d'utilité des autres agents.

7.2.1 Mesure des comportements individuels

Une mesure d'utilité a été établie à partir de l'équation de Bellman 2.3 afin de quantifier la valeur du comportement (ou plan d'action) d'un agent. Elle évalue la politique individuelle d'un

agent à partir d'un état s_i et associe donc à chaque état s_i une valeur traduisant l'utilité espérée de l'agent lorsqu'il est dans s_i. Cette valeur est égale à la somme :

- de la récompense obtenue par l'agent pour être dans l'état s_i (gain immédiat)
- et du gain que l'agent peut espérer obtenir à partir de cet état (utilité espérée).

Le calcul de l'utilité associée à un état s_i est alors réalisé à partir de l'équation suivante :

$$V(s_i) = \overbrace{R(s_i)}^{\text{Gain immédiat}} + \overbrace{max_{E(t_{i+1},st),st \geq et(I')_{t_i}}(V')}^{\text{Utilité espérée}} \tag{7.1}$$

Si $s_i = [t_i, [st(I')_{t_i}, et(I')_{t_i}], r_{t_i}]$ (état de succès), $R(s_i) = \mathcal{R}(t_i)$. Sinon (état d'échec partiel), $R(s_i) = 0$. En effet, lorsqu'un agent arrive dans un état d'échec partiel, l'exécution de la tâche a échoué, l'agent ne reçoit donc aucune récompense.

Lors de l'exécution des tâches, un agent $\mathcal{A}g_i$ passe par au plus un seul état de succès associé à une tâche t_i donnée. L'agent ne peut en effet pas passer par deux états s_i et s'_i tels que $s_i = [t_i, *, *]$, $s'_i = [t_i, *, *]$ et $s_i \neq s'_i$ puisque l'agent arrive dans un état de succès associé à t_i lorsqu'il termine l'exécution de cette tâche et seulement dans ce cas. La tâche t_i étant exécutée une seule fois, l'agent ne peut arriver qu'une seule fois dans un état de succès associé à t_i. La récompense associée à la tâche t_i ne peut donc pas être touchée plusieurs fois par l'agent.

L'utilité espérée V' doit tenir compte des différents états dans lesquels l'agent peut arriver en exécutant la tâche t_{i+1}. Nous avons montré au chapitre précédent qu'il existait trois grands types de transitions possibles (succès, échec partiel, échec total). V' peut donc être décomposée en trois termes :

$$V' = V_{suc} + V_{PPC} + V_{fail} \tag{7.2}$$

où V_{suc} est l'utilité espérée de l'agent lorsque l'exécution de la tâche réussie, V_{PPC} désigne l'utilité espérée de l'agent lorsque l'exécution de la tâche échoue partiellement et V_{fail} représente l'utilité espérée lorsque l'exécution de la tâche échoue totalement.

- *Exécution réussie :*

$$V_{suc} = \overbrace{P^r_{enough}(Pred(t_{i+1})) \prod_{a \in P_{red}(t_{i+1}) \setminus t_i} \sum_{st' \leq st} P^a_{\mathcal{ET}}(st'|et(I')_{t_i})}^{\text{Probabilité de succès}}$$
$$\overbrace{\cdot \sum_{\Delta_r^{i+1}|r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1}|st+\delta_c^{i+1} \leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}}$$
$$\cdot V([t_{i+1}, [st, st + \delta_c^{i+1}], r_{t_i} - \Delta_r^{i+1}])$$

où $V([t_{i+1}, [st, st+\delta_c^{i+1}], r_{t_i} - \Delta_r^{i+1}])$ désigne l'utilité de l'état $[t_{i+1}, [st, st+\delta_c^{i+1}], r_{t_i} - \Delta_r^{i+1}]$ dans lequel arrive l'agent lorsque l'exécution de la tâche consomme Δ_r^{i+1} ressources et dure δ_c^{i+1} unités de temps.

– *Échec partiel :*

$$V_{PPC} = \overbrace{P_{PPC}}^{\text{Probabilité d'échouer partiellement}} \cdot V([t_i, [st, st+1], et(I')_{t_i}, r_{t_i} - \Delta r'])$$

– *Échec total :*

$$V_{fail} = \overbrace{P_{MR} + P_{TT} + P_{DM}}^{\text{Probabilité d'échouer totalement}} \cdot V([failure_{t_{i+1}}, *, *])$$

$V([failure_{t_{i+1}}, *, *])$ désigne la valeur de l'état d'échec total associé à t_{i+1}. A chaque tâche est associé un unique état d'échec total dans lequel l'agent arrive quelle que soit la cause de son échec. La valeur de l'état est alors indépendante des ressources à disposition de l'agent ainsi que de la date de l'échec. Nous désignons donc l'état d'échec total associé à t_{i+1} par : $[failure_{t_{i+1}}, *, *]$.

$V([failure_{t_{i+1}}, *, *])$ est définie par l'équation suivante :

$$V([failure_{t_{i+1}}, *, *]) = -\mathcal{R}(t_{i+1}) - \sum_{t_{suiv} \in \mathcal{T}_{suiv}(t_{i+1})} \mathcal{R}(t_{suiv})$$

où $-\mathcal{R}(t_{i+1})$ correspond à la pénalité immédiate donnée à l'agent $\mathcal{A}g_i$ pour avoir échoué totalement lors de l'exécution de la tâche t_{i+1} et $-\sum_{t_{suiv} \in \mathcal{T}_{suiv}(t_{i+1})} \mathcal{R}(t_{suiv})$ représente la perte future de l'agent, elle correspond à la somme des récompenses qu'il aurait pu obtenir s'il avait eu la possibilité d'exécuter toutes les tâches $\mathcal{T}_{suiv}(t_{i+1})$ qu'il lui restait à réaliser.

A partir de l'équation 7.1, nous sommes en mesure de calculer l'utilité espérée de l'agent $\mathcal{A}g_i$. Nous allons donc à présent nous intéresser à l'estimation des répercussions d'une décision sur les autres agents. Nous allons en particulier mesurer la perte provoquée sur les autres agents lorsque $\mathcal{A}g_i$ décide de commencer l'exécution d'une tâche t_{i+1} à st.

7.2.2 Mesure des influences entre agents

Le coût occasionné exprime les bénéfices qui aurait pu être faits en choisissant une autre alternative. Comme nous l'avons expliqué précédemment, cette notion va nous permettre de coordonner les décisions des agents et ainsi d'obtenir un comportement coopératif de leur part. Pour ce faire, nous devons être en mesure de calculer le coût occasionné par un agent $\mathcal{A}g_i$ lorsqu'il décide de commencer l'exécution d'une tâche t_{i+1} à st. Il nous est donc nécessaire de déterminer la perte générée par cette décision sur les agents $\mathcal{A}g_j$ tels que $j \neq i$.

Considérons deux tâches t_{i+1} et t_j exécutées respectivement par les agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$ et telles que t_j soit un successeur de t_{i+1}. La figure 7.1 présente deux exécutions possibles de t_{i+1}. La première commence à $st1$ et termine à $et1$, la seconde commence à $st2$ et termine à

$et2$ ($et2 \geq et1 \geq LB_j$). Les bornes LB_j et UB_j sur les dates de début de t_j sont également représentées. Lorsque t_{i+1} termine à $et1$, t_j peut commencer dans l'intervalle $[LB_j, UB_j]$. Nous désignerons par st_j la meilleure date de début pour t_j dans cet intervalle. En revanche, si t_{i+1} termine à $et2$, l'intervalle des dates de début possibles pour t_j est restreint à $[LB_j + \Delta t, UB_j]$ où $\Delta t = et2 - LB_j$. st'_j désigne alors la meilleure date de début de t_j dans l'intervalle $[LB_j + \Delta t, UB_j]$.

Si $st_j = st'_j$, le fait que t_{i+1} termine plus tard ne change pas la meilleure date de début pour t_j, le coût occasionné est par conséquent nul.

En revanche, si $st_j \neq st'_j$, l'utilité espérée V^{st_j} de $\mathcal{A}g_j$ lorsque t_j commence à st est nécessairement supérieure à l'utilité espérée $V^{st'_j}$ de $\mathcal{A}g_j$ lorsque t_j commence à st'_j sinon, nous aurions $st_j = st'_j$. Le fait que t_{i+1} finisse à $et2$ provoque donc une perte en utilité espérée pour $\mathcal{A}g_j$ correspondant au coût occasionné par $\mathcal{A}g_i$ sur $\mathcal{A}g_j$.

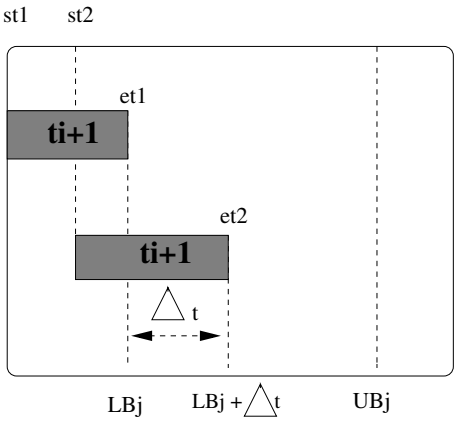


FIG. 7.1 – Coût occasionné

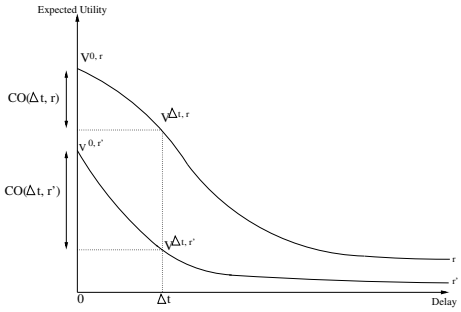


FIG. 7.2 – Influence des ressources et du retard sur le coût occasionné

La politique de chaque agent étant dépendante de la quantité de ressources disponible, cette dernière influence nécessairement l'utilité espérée et par conséquent la mesure du coût occasionné. La figure 7.2 illustre l'influence du retard et des ressources sur l'utilité espérée et le coût occasionné d'un agent. Lorsqu'il reste à l'agent $\mathcal{A}g_j$ très peu de ressources, la probabilité que l'exécution de sa prochaine tâche t_j échoue par manque de ressources, est très forte et ce, quelle que soit la date de début de t_j. Ainsi, retarder l'exécution de t_j a peu de répercussions puisque de toute façon l'agent a peu de chances de réussir l'exécution de sa tâche. En revanche, lorsqu'il reste à l'agent juste assez de ressources pour exécuter ses tâches restantes, retarder t_j peut au contraire avoir une grande influence. Cela risque en effet de conduire l'agent à un échec partiel et par conséquent consommer des ressources qui lui étaient indispensables par la suite. Dans de telles circonstances, le coût occasionné provoqué en retardant l'exécution de t_j est beaucoup plus

élevé que dans le cas précédent.

Les contraintes temporelles influencent par ailleurs indirectement le coût occasionné. En effet, quelles que soient les ressources disponibles, si retarder une tâche t_j ne permet plus de respecter les contraintes temporelles de la tâche alors, le coût occasionné provoqué sur $\mathcal{A}g_j$ sera élevé puisque ce retard le conduit à un échec total. De manière plus générale, soient Δt et $\Delta t'$ deux valeurs de retard telles que $\Delta t < \Delta t'$. Pour une même quantité de ressources r_{t_j}, si retarder t_j de Δt n'empêche pas l'agent $\mathcal{A}g_j$ de respecter les contraintes temporelles de t_j, mais que retarder t_j de $\Delta t'$ entraîne une violation des contraintes de t_j, alors $OC(\Delta t, r_{t_j}) < OC(\Delta t', r_{t_j})$.

Une valeur de coût occasionné doit donc être définie pour chaque couple [retard, ressources]. Pour chaque date de début st_j d'une tâche t_j et chaque taux de ressources r_{t_j}, l'utilité espérée de l'agent $V_{t_j}^{\Delta t, r_{t_j}}$ est alors déterminée ($\Delta t = st_{t_j} - LB_{t_j}$). Il est ensuite possible de déduire, à partir de ces valeurs, le coût occasionné $OC_{t_j}(\Delta t, r_{t_j})$ tel que :

$$OC_{t_j}(\Delta t, r_{t_j}) = V_{t_j}^{0, r_{t_j}} - V_{t_j}^{\Delta t, r_{t_j}} \tag{7.3}$$

$V_{t_j}^{0, r_{t_j}}$ correspond à l'utilité espérée de l'agent $\mathcal{A}g_j$ lorsque l'exécution de t_j peut commencer dans l'intervalle $[LB_j, UB_j]$ avec r_{t_j} ressources. $V_{t_j}^{\Delta t, r_{t_j}}$ est l'utilité espérée de l'agent $\mathcal{A}g_j$ lorsque l'exécution de t_j peut commencer dans l'intervalle $[LB_j + \Delta t, UB_j]$ avec r_{t_j} ressources. Ces valeurs sont calculées par l'équation suivante :

$$V_{t_j}^{\Delta t, r_{t_j}} = \sum_{\Delta_r^j} \sum_{\delta_c^j} P_r(\Delta_r^j) P_c(\delta_c^j).V[t_j, [st(LB_{t_j} + \Delta t), st(LB_{t_j} + \Delta t) + \delta_c^j], r_{t_j} - \Delta_r^j - \Delta r' \times nbEP]$$

où $st(LB_{t_j} + \Delta t)$ désigne la meilleure date de début pour la tâche t_j dans l'intervalle $[LB_j + \Delta t, UB_j]$. $nbEP$ est le nombre d'échecs partiels induits par le retard et déduit à partir de la politique de l'agent $\mathcal{A}g_j$. Δ_r^j est la consommation de la tâche t_j, $\Delta r'$ correspond à la consommation de ressources d'un échec partiel et δ_c^j désigne la durée d'exécution de t_j.

Corollaire 2 *Le coût occasionné est toujours positif ou nul.*

Preuve : Afin de démontrer ce corollaire, nous allons procéder à un raisonnement par l'absurde. Supposons que le coût occasionné puisse être négatif. D'après l'équation 7.3, nous avons alors $V_{t_j}^{0, r_{t_j}} < V_{t_j}^{\Delta t, r_{t_j}}$ (fonction décroissante en fonction de Δt).

Soit st_j la meilleure date de début de la tâche t_j appartenant à l'intervalle $[LB_j, UB_j]$. $V_{t_j}^{0, r_{t_j}}$ correspond l'utilité de l'agent lorsque l'exécution de t_j commence à st_j. Soit st'_j la meilleure date de début de la tâche t_j appartenant à l'intervalle $[LB_j + \Delta t, UB_j]$. $V_{t_j}^{\Delta t, r_{t_j}}$ correspond à l'utilité de l'agent lorsque l'exécution de t_j commence à st'_j.

Si st_j appartient à $[LB_j + \Delta t, UB_j]$ alors elle correspond également à la meilleure date de début de cet intervalle. ($[LB_j + \Delta t, UB_j] \subseteq [LB_j, UB_j]$). Nous pouvons en déduire que $st_j = st'_j$, d'où $V_{t_j}^{0, r_{t_j}} = V_{t_j}^{\Delta t, r_{t_j}}$. Ceci contredit notre hypothèse de départ.

Si st_j n'appartient pas à l'intervalle $[LB_j + \Delta t, UB_j]$, la meilleure date de début st'_j appartenant à l'intervalle $[LB_j + \Delta t, UB_j]$ est telle que $st_j \neq st'_j$ et $V_{t_j}^{0,r_{t_j}} > V_{t_j}^{\Delta t,r_{t_j}}$. En effet, si $V_{t_j}^{0,r_{t_j}}$ était inférieure ou égale à $V_{t_j}^{\Delta t,r}$, st'_j correspondrait à la meilleure date de début appartenant à $[LB_j + \Delta t, UB_j]$ et nous aurions $st_j = st'_j$. Nous sommes donc à nouveau en contradiction avec notre hypothèse de départ.

Nous pouvons par conséquent déduire que $V_{t_j}^{0,r_{t_j}} \geq V_{t_j}^{\Delta t,r_{t_j}}$. D'après l'équation 7.3, cette inégalité entraîne : $OC_{t_j}(\Delta t, r_{t_j}) \geq 0$ quels que soient t_j, Δt et r_{t_j}. □

Le coût occasionné $OC_{t_j}(\Delta t, r_{t_j})$ provoqué par un agent $\mathcal{A}g_i$ lorsqu'il décide de commencer l'exécution d'une tâche t_i à st traduit donc toujours une perte d'utilité correspondant à la diminution d'utilité de l'agent $\mathcal{A}g_j$ due au fait que sa tâche t_j ne pourra commencer avant $LB_j + \Delta t$. Remarquons que cette perte est indépendante de l'agent ayant provoqué le retard.

7.2.3 Calcul de la politique

Nous venons d'établir deux mesures. La première définit la valeur d'un état s_i d'un agent $\mathcal{A}g_i$ à partir du gain immédiat de $\mathcal{A}g_i$ et de l'utilité espérée V' des tâches qu'il lui reste à exécuter. Pour un agent égocentré, c'est-à-dire cherchant à maximiser sa propre utilité et non celle du système, la meilleure action à exécuter à partir de s_i est donc l'action maximisant l'utilité espérée V'.

La seconde mesure détermine la perte en utilité provoquée sur un agent $\mathcal{A}g_j$ lorsque l'exécution d'une de ses tâches t_j est retardée de Δt unités de temps. En effet, lorsqu'un agent $\mathcal{A}g_i$ décide de commencer l'exécution d'une tâche t_i à st, il provoque un retard plus ou moins important sur l'exécution des prochaines tâches (tâches accessibles à partir de t_i dans le graphe de la mission). L'impact de ce retard Δt sur les autres agents est mesuré par le coût occasionné traduisant la perte en utilité provoquée par Δt.

Afin de maximiser l'utilité du groupe et non sa propre utilité, chaque agent doit tenir compte des répercussions de ses décisions sur les autres agents. La meilleure action qu'un agent puisse exécuter à partir d'un état s_i résulte alors d'un compromis entre :

– l'utilité espérée de l'agent V' calculée par l'équation 7.2,

– le coût occasionné provoqué sur les autres agents.

Ainsi, la politique de l'agent $\mathcal{A}g_i$ à partir de l'état s_i est calculée par l'équation suivante :

$$\pi_i(s_i) = arg\_max_{E(t_{i+1},st),st \geq et(I')_{t_i}} \Big(\overbrace{V'}^{\text{Utilité espérée}} - \overbrace{OC(t_{i+1},st)}^{\text{Coût occasionné}} \Big) \tag{7.4}$$

L'utilité espérée tient compte de l'impact de la décision de l'agent $\mathcal{A}g_i$ sur toutes ses tâches à venir. Le coût occasionné traduit quant à lui l'influence de cette décision sur les tâches accessibles

à partir de t_{i+1} et exécutées par un agent $\mathcal{A}g_j$ tel que $j \neq i$.

Considérons le graphe de la figure 6.1. La meilleure date de début pour la tâche B est déterminée en tenant compte de l'utilité espérée de l'agent 2 lors de l'exécution des tâches B, F et G, et du coût occasionné provoqué sur les agents 1 et 3. Ce dernier correspond à la perte en utilité des agents 1 et 3 due au retard provoqué par l'exécution de B sur les tâches D et E.

L'utilisation de l'équation 7.4 pour déterminer la politique de chaque agent, garantit que chacun suivra une politique coopérative. En effet, l'exécution d'une tâche t_i sera retardée de Δt si et seulement si, le gain de l'agent $\mathcal{A}g_i$ exécutant t_i (gain en utilité espérée) est supérieur aux pertes provoquées par ce retard sur les autres agents.

7.3 Estimation de l'influence d'une action

Nous avons précédemment défini le coût occasionné comme la perte en utilité d'un agent $\mathcal{A}g_j$ lorsque l'exécution d'une de ses tâches t_j est retardée de Δt. Nous nous plaçons cependant dans un contexte où l'exécution des tâches est incertaine et par conséquent où les durées d'exécution sont stochastiques. Lorsqu'un agent commence l'exécution d'une tâche t_i à st, nous ne pouvons pas déduire de manière certaine le retard Δt provoqué sur une tâche t_j. En effet, suivant la durée d'exécution t_i, un retard plus ou moins important sera induit.

Nous allons, dans cette section, décrire comment tenir compte de cette incertitude sur l'amplitude du retard. Nous montrerons comment calculer le coût occasionné provoqué sur l'agent $\mathcal{A}g_j$ lorsque l'exécution de t_i commence à st. Nous serons alors en mesure d'établir le lien entre le coût occasionné $OC(t_{i+1}, st)$ utilisé dans l'équation 7.3 et la valeur de coût occasionné $OC_{t_j}(\Delta t, r_{t_j})$ calculée par l'équation 7.2.

7.3.1 Estimation du retard

Nous avons précédemment décrit comment calculer le coût occasionné provoqué par un retard Δt et avons également défini les différentes probabilités de transition entre états lors de l'exécution d'une tâche t_{i+1}. Cet ensemble d'informations va nous permettre de déduire le coût occasionné provoqué par un agent $\mathcal{A}g_i$ lorsque celui-ci commence l'exécution d'une tâche t_{i+1} à st, soit $OC(t_{i+1}, st)$.

Pour ce faire, chacune des transitions possibles de l'agent doit être considérée. A partir des probabilités des transitions, les probabilités sur les dates de fin de la tâche peuvent ainsi être calculées. Le coût occasionné provoqué par un agent $\mathcal{A}g_i$ lorsqu'il décide de commencer l'exécution d'une tâche t_{i+1} à st est tel que :

$$OC(t_{i+1}, st) = \overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a \in P_{red}(t_{i+1})-t_i} \sum_{s \in ET(a)|s \leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}} \tag{7.5}$$

$$\overbrace{. \sum_{\Delta_r^{i+1}|r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1}|st+\delta_c^{i+1} \leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}} \cdot \sum_{\mathcal{A}g_j \in \text{other\_agents}} EOC_{\mathcal{A}g_j,t_{i+1}}(et_{i+1})$$

$$+ \overbrace{(P_{MR} + P_{TT} + P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j \in \text{other\_agents}} EOC_{\mathcal{A}g_j,t_{i+1}}(fail)$$

$$+ \overbrace{P_{PPC}}^{\text{Probabilité d'échouer partiellement}} \cdot OC(t_{i+1}, next\_start)$$

où $EOC_{\mathcal{A}g_j,t_{i+1}}(et_{i+1})$ correspond au coût occasionné provoqué par l'agent $\mathcal{A}g_i$ sur l'agent $\mathcal{A}g_j$ lorsque l'exécution de t_{i+1} s'achève à et_{i+1}.

$OC(t_{i+1}, st)$ est défini en considérant chacune des exécutions possibles de t_{i+1} lorsque l'agent $\mathcal{A}g_i$ tente de l'exécuter à st. Lorsque $\mathcal{A}g_i$ échoue partiellement, le coût occasionné est déterminé par $OC(t_{i+1}, next\_start)$ où $next\_start$ correspond à la prochaine date à laquelle l'agent tentera d'exécuter t_{i+1}. Lorsque l'exécution de la tâche t_{i+1} échoue totalement, la tâche t_j ne pourra être exécutée, le coût occasionné $EOC_{\mathcal{A}g_j,t_{i+1}}(fail)$ traduisant la perte de l'agent $\mathcal{A}g_j$ est donc égal à la somme des récompenses que $\mathcal{A}g_j$ ne pourra obtenir, soit :

$$EOC_{\mathcal{A}g_j,t_{i+1}}(fail) = OC_{t_j}(fail) = \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})\Big(V^0_{t_j,r_{t_j}} - V([failure_{t_j}, *, *])\Big)$$

7.3.2 Coût occasionné espéré

Connaissant la date de fin d'exécution de t_{i+1}, il nous reste à estimer le retard provoqué sur les autres agents ce qui permettra de définir le coût occasionné espéré en fonction des valeurs $OC(t_{i+1}, st)$ précédemment calculées.

Si l'agent $\mathcal{A}g_i$ termine t_{i+1} à $et_{t_{i+1}}$, il n'est pas certain que la tâche la plus proche t_j exécutée par $\mathcal{A}g_j$ va commencer à $et_{t_{i+1}}$. Si t_j n'est pas un successeur direct de t_{i+1}, il existe d'autres tâches t_k, entre t_{i+1} et t_j, dans le graphe de la mission. Le retard provoqué par t_{i+1} peut alors être compensé ou accentué par les tâches t_k.

De manière à tenir compte de l'incertitude du retard induit sur t_j, nous avons défini une mesure appelée « coût occasionné espéré » estimant pour chaque date de fin et_{i+1} de t_{i+1}, la probabilité que t_j commence à st_j. Une fois la date de début st_j de t_j établie, le retard sur l'exécution de la tâche t_j peut être calculé et le coût occasionné provoqué par t_{i+1} est déterminé.

Ce dernier est alors obtenu par l'équation suivante :

$$EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}}) = max_{t_k \in Suc(t_{i+1})} EOC_{t_k}(et_{t_{i+1}} - LB_k, t_j)$$

où t_j est la prochaine action la plus proche dans le graphe qui va être exécutée par $\mathcal{A}g_j$, c'est-à-dire la première tâche de $\mathcal{A}g_j$ sur laquelle le retard provoqué par t_{i+1} aura un impact. Par exemple, sur le graphe de la figure 6.1, la tâche la plus proche de B exécutée par l'agent 3 est D et celle exécutée par l'agent 1 est E. $EOC_{t_k}(et_{t_{i+1}} - LB_k, t_j)$ désigne le coût occasionné attendu provoqué sur t_j lorsque la tâche t_k est retardée de $et_{t_{i+1}} - LB_k$.

Si $t_j = t_k$

$$EOC_{t_j}(et_{t_{i+1}} - LB_j, t_j) = \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) \times OC_j(et_{t_{i+1}} - LB_j, r_{t_j})$$

Sinon

$$EOC_{t_k}(et_{t_{i+1}} - LB_k, t_j) = \sum_{r_{t_k}} P_{ra}^{t_k}(r_{t_k}) \sum_{\delta_c^k} P_c(\delta_c^k) max_{t_l \in Suc(t_k)} EOC_{t_l}(st^* + \delta_c^k - LB_l, t_j)$$

où st^* est la date de début à laquelle l'exécution de t_k va commencer lorsque l'agent exécutant la tâche t_k dispose de r_{t_k} ressources avant l'exécution de t_k et qu'il ne peut commencer son exécution avant $et_{t_{i+1}}$. Précisons que si $et_{t_{i+1}} > UB_k$ alors, $EOC_{t_k}(et_{t_{i+1}} - LB_k, t_j) = OC_j(fail)$.

Afin de déterminer la valeur de $EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}})$, nous considérons le coût occasionné espéré provoqué sur t_j lorsque les successeurs t_k de t_{i+1} sont retardés. Ce coût occasionné est lui même calculé en considérant le coût occasionné espéré provoqué sur t_j par les successeurs t_l de t_k. Pour ce faire, nous déterminons à partir de la politique de l'agent exécutant t_k, la date de début st^* de t_k. En tenant compte des durées d'exécution de la tâche, nous pouvons déduire ses dates de fin et par conséquent le retard provoqué sur ses successeurs t_l.

Par récurrence, nous arrivons ainsi à la tâche t_j dont le coût occasionné peut être déterminé à partir des valeurs $OC_j(\Delta t, r_{t_j})$, puisque nous connaissons alors le retard effectivement provoqué sur la tâche.

La valeur du coût occasionné provoqué sur un agent $\mathcal{A}g_j$ lorsque l'exécution de sa plus proche tâche t_j est retardée de Δt étant positive ou nulle (cf. corollaire 2), le coût occasionné espéré $EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}})$ provoqué par une tâche t_{i+1} lorsqu'elle termine à $et_{t_{i+1}}$ est positif ou nul. Nous en déduisons que $OC(t_{i+1}, st)$ est également positif ou nul. Ce terme utilisé dans l'équation 7.4 traduira donc toujours un coût et pénalisera l'agent lorsqu'il retardera l'exécution des tâches des autres agents.

Le calcul du coût occasionné espéré que nous venons de décrire détermine de manière précise le retard induit sur les autres agents. Ainsi une mesure précise du coût occasionné est obtenue et

utilisée afin de décider quand commencer l'exécution d'une tâche t_{i+1}. L'utilisation d'une mesure approchée du coût occasionné précis est également envisageable. Néanmoins, elle risque, dans certains cas, de conduire à une prise de décision inadaptée en raison de la mauvaise estimation de ses répercussions sur les autres agents. C'est ce que nous allons illustrer en présentant différentes méthodes d'estimation du coût occasionné sur lesquelles nous avons travaillé avant d'aboutir à la mesure qui vient d'être présentée.

7.4 Autres mesures du coût occasionné

Différentes mesures approchant le coût occasionné précis peuvent être envisagées. Leur calcul est généralement plus direct que celui du coût occasionné espéré, mais elles peuvent conduire à une estimation inexacte de l'influence d'une décision sur les autres agents et par conséquent à des comportements non coopératifs. Nous allons présenter plusieurs de ces méthodes et discuter les cas où elles s'avèrent inappropriées.

7.4.1 Indépendance vis à vis des ressources

Une première approximation du coût occasionné précis est obtenue en occultant l'influence des ressources qui sont à disposition des agents. Le coût occasionné provoqué par le retard d'une tâche t_{i+1} sur un agent $\mathcal{A}g_j$ est alors calculé en supposant que les ressources de l'agent $\mathcal{A}g_j$ sont maximales. Une telle approximation du coût occasionné est décrite dans [Beynier et Mouaddib, 2005a].

Les équations permettant d'établir cette approximation diffèrent très peu de celles que nous avons précédemment présentées. Les modifications affectent essentiellement le calcul de $V_{t_j}^{\Delta t}$ qui est défini par l'équation suivante :

$$V_{t_j}^{\Delta t} = \sum_{\delta_c^j} P_c(\delta_c^j).V[t_j, [st(LB_{t_j} + \Delta t), st(LB_{t_j} + \Delta t) + \delta_c^j], r_{t_j}^{max}] \tag{7.6}$$

où $st(LB_{t_j}+\Delta t)$ est la meilleure date de début pour t_j appartenant à l'intervalle $[LB_j+\Delta t, UB_j]$.

$[t_j, [st_{t_j}, st_{t_j} + \delta_c^j], r_{t_j}^{max}]$ décrit les états pouvant être atteints lorsque l'exécution de t_j commence à $st_{t_j} \in [LB_j + \Delta t, UB_j]$ et dure δ_c^j unités de temps. Si $st_{t_j} + \delta_c^j > LET$, l'exécution échoue totalement, nous obtenons alors :

$$V[t_j, [st_{t_j}, st_{t_j} + \delta_c^j], r_{t_j}^{max}] = V([failure_{t_j}, *, *])$$

Il en suit une suppression de toute référence aux ressources dans les équations établissant la valeur du coût occasionné, d'où :

$$OC_{t_j}(\Delta t) = V_{t_j}^0 - V_{t_j}^{\Delta t}$$

et

$$EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}}) = \sum_{st_{t_j} \in ST(t_j)} P^r_{enough}(Pred(t_j)).P_{\mathcal{ST}}(st_{t_j}, et_{t_{i+1}}, t_{i+1}) \cdot OC_{t_j}(st_{t_j} - LB) + (1 - (\sum_{st \in ST(t_j)} P_{\mathcal{ST}}(st_{t_j}, et_{t_{i+1}}, t_{i+1}) + P^r_{enough}(Pred(t_j)))) \cdot OC_{t_j}(fail)$$

Une telle estimation du coût occasionné réduit le nombre de valeurs OC_{t_j} à calculer puisqu'une valeur de coût occasionné doit être déterminée pour chaque retard possible au lieu de chaque couple [retard, ressources]. Néanmoins, suivant les ressources réellement à disposition de l'agent, cette estimation peut mener à une sur-estimation ou à une sous-estimation de l'influence d'une action.

Si l'agent $\mathcal{A}g_j$ dispose par exemple de très peu de ressources, de sorte qu'il ne peut même pas exécuter la tâche t_j, le coût occasionné calculé par cette méthode d'approximation risque de sur-estimer le coût occasionné précis. En effet, le coût occasionné précis provoqué sur l'agent $\mathcal{A}g_j$ est très faible puisque quel que soit le retard, la tâche t_j échouera par manque de ressources. En revanche, si le coût occasionné est calculé en supposant que l'agent dispose de $r^{max}_{t_j}$ ressources et que celles-ci lui permettent d'exécuter t_j, retarder la tâche t_j suscitera une perte en utilité non envisagée lors du calcul du coût occasionné précis. Le coût occasionné approché sur-estimera par conséquent le coût occasionné précis.

A l'inverse, si l'agent dispose d'une quantité r_{t_j} de ressources juste assez suffisante pour exécuter ses tâches, considérer que l'agent dispose de plus de ressources ($r^{max}_{t_j} > r_{t_j}$) mène à une sous-estimation du coût occasionné précis. En effet, lorsque l'agent a juste assez de ressources pour exécuter ses tâches, tout retard est susceptible de le conduire à un échec partiel qui par sa consommation le mènera dans une situation où il ne pourra plus exécuter ses tâches (manque de ressources). Dans une telle situation, le coût occasionné pour un retard donné est plus important que dans le cas où l'agent dispose d'une quantité $r^{max}_{t_j}$ de ressources permettant éventuellement quelques échecs partiels.

Dans les cas où les agents disposent de bien assez de ressources pour exécuter leurs tâches et peuvent ainsi faire face aux consommations d'éventuels échecs partiels, ne pas tenir compte des ressources lors du calcul du coût occasionné importe peu. En effet, les valeurs ainsi déterminées constituent une bonne approximation du coût occasionné précis puisque celui-ci est peu influencé par les ressources à disposition de l'agent. Les erreurs d'approximation se font donc essentiellement sentir lorsque les agents sont « justes » en ressources ou manquent de ressources, elles risquent alors de conduire à des prises de décision inadaptées.

7.4.2 Modification de l'équation de Bellman

Au lieu de considérer le coût occasionné provoqué sur les autres agents lorsque $\mathcal{A}g_i$ commence l'exécution de t_{i+1} à st, il est possible d'envisager le coût occasionné provoqué sur les successeurs directs de t_{i+1} lorsque celle-ci se termine à et. Il n'est alors pas nécessaire de gérer d'éventuelles augmentations ou diminutions du retard provoqué initialement par $\mathcal{A}g_i$. En effet, puisqu'il n'existe pas de tâche entre t_{i+1} et ses successeurs directs, le retard provoqué par $\mathcal{A}g_i$ à la fin de l'exécution de t_{i+1} correspond au retard induit sur les successeurs directs de t_{i+1}. Le recours au coût occasionné espéré n'est par conséquent plus justifié.

Néanmoins, retarder l'exécution d'une tâche t_{i+1} n'a pas uniquement des répercussions sur le successeurs directs de t_{i+1}, toutes les tâches accessibles à partir de t_{i+1} sont concernées. L'influence de la date de début de t_{i+1} sur ces tâches peut alors être gérée par une relation de récurrence. Pour ce faire, le coût occasionné provoqué sur les successeurs de t_{i+1} doit être calculé en tenant compte du coût occasionné sur les successeurs des successeurs de t_{i+1}, et ainsi de suite. Cette méthode de calcul, décrite dans [Beynier et Mouaddib, 2004a, Beynier et Mouaddib, 2004d], utilise l'équation de Bellman transformée suivante :

$$V(s_i) = \overbrace{R(s_i)}^{\text{gain immédiat}} - \overbrace{\sum_{t_j \in Succ(t_i)} OC_{t_j}(\Delta t)}^{\text{coût occasionné}} + \overbrace{max_{E(t_{i+1},st),st \geq et(I')_{t_i}}(V')}^{\text{gain espéré}}$$

Contrairement aux méthodes précédemment décrites, la valeur d'un état s_i n'est pas déterminée par une équation de Bellman usuelle. En effet, le gain immédiat obtenu par l'agent $\mathcal{A}g_i$ lorsqu'il est dans un état s_i est pondéré par le coût occasionné sur les successeurs de la tâche t_i lorsque cette dernière s'achève avec un retard Δt. Le coût occasionné est calculé par une équation similaire aux équations qui ont été présentées jusqu'ici :

$$OC_{t_j}(\Delta t) = V^0_{t_j} - V^{\Delta t}_{t_j} \tag{7.7}$$

Le coût occasionné $OC_{t_j}(\Delta t)$ provoqué sur un successeur t_j correspond donc à la différence d'utilité de l'agent $\mathcal{A}g_j$ lorsque t_j est retardée de Δt. L'utilité de $\mathcal{A}g_j$ est elle même calculée en considérant le coût occasionné sur les successeurs de t_j, qui prend lui même en compte le coût occasionné sur les successeurs des successeurs de t_j, et ainsi de suite. $OC_{t_j}(\Delta t)$ tient alors compte du coût occasionné suscité sur tous les successeurs directs ou indirects de t_j.

Cette méthode de calcul présente néanmoins quelques inconvénients. Elle ne tient tout d'abord pas compte des ressources à disposition de l'agent ce qui nous ramène aux difficultés exposées précédemment. Par ailleurs, suivant le configuration du graphe, il est possible de comptabiliser plusieurs fois le coût occasionné provoqué sur une même tâche t_j. En effet, s'il existe à partir d'une tâche t_i plusieurs chemins pouvant mener à la tâche t_j alors, le coût occasionné suscité par t_i sur t_j sera pris en compte autant de fois qu'il existe de chemins. Reprenons le graphe de

la figure 6.1. Le coût occasionné provoqué par B sur D est calculé à partir de l'utilité de l'agent 3 lorsque ce dernier exécute D. Cette dernière est déterminée en considérant le coût occasionné sur E et sur F. L'utilité de l'agent 1 lorsque E est exécutée tient compte du coût occasionné sur G, donc de l'utilité de l'agent 2 lorsque celui-ci exécute G. L'utilité de l'agent 2 lorsqu'il exécute E est également calculée en considérant l'utilité de l'exécution de G. La tâche G est donc comptabilisée deux fois.

Cette comptabilisation multiple du coût occasionné sur une même tâche conduit alors à une sur-estimation de la valeur du coût occasionné précis. Afin de résoudre ce problème, il est nécessaire d'extraire le coût occasionné du calcul de l'utilité espérée et ainsi envisager le coût occasionné indépendamment de l'utilité espérée. Cette méthode suppose alors de considérer le coût occasionné provoqué sur les autres agents, au lieu du coût occasionné engendré sur les successeurs. Nous nous ramenons donc à la méthode de calcul du coût occasionné présentée précédemment.

Conclusion

Dans ce chapitre, nous nous sommes intéressée à la résolution du problème de planification multi-agent défini précédemment. Afin de considérer des problèmes de taille conséquente, nous avons proposé d'exploiter la décomposition du problème réalisée lors de sa modélisation. Ainsi, nous cherchons à résoudre un ensemble de MDPs représentant chacun le problème décisionnel d'un agent.

Afin d'obtenir un comportement coopératif permettant de maximiser l'utilité collective, nous avons montré que les MDPs ne pouvaient être résolus totalement indépendamment les uns des autres. Il est en effet nécessaire que chaque agent tienne compte de l'influence de ses décisions sur les autres agents. A partir de cette constatation, nous avons introduit la notion de coût occasionné permettant de rendre compte des pertes provoquées par un agent sur les autres membres du système, lorsque celui-ci décide d'exécuter une action a_i à partir d'un état s_i. Toute prise de décision résulte alors d'un compromis entre l'utilité espérée de l'agent et la perte provoquée sur les autres agents. Nous avons pu établir une mesure précise du coût occasionné et avons également présenté différentes approximations de cette mesure. Nous avons alors discuté les cas où leur utilisation pouvait mener à des décisions inadaptées.

Étant donné un agent $\mathcal{A}g_i$, nous sommes désormais capable de décider quelle action celui-ci doit exécuter à partir d'un quelconque état s_i. Pour ce faire, il est néanmoins nécessaire de connaître le coût occasionné provoqué sur les autres agents ce qui suppose que les politiques d'exécution des tâches influencées par $\mathcal{A}g_i$ soient connues. Afin d'appliquer le principe de décision exposé dans ce chapitre, il est indispensable de développer un algorithme de calcul des politiques permettant de disposer de ces informations lorsque la politique de $\mathcal{A}g_i$ en s_i est déterminée.

Chapitre 8

Approximation par révision de la politique initiale

Nous avons établi, au chapitre précédent, un ensemble d'équations permettant de décider quelle action exécuter à partir d'un état s_i. Nous souhaitons à présent définir la politique complète de chaque agent, c'est-à-dire associer à tout état s_i de tout agent $\mathcal{A}g_i$, une action. Pour ce faire, nous allons, dans ce chapitre, proposer un algorithme utilisant les équations précédemment définies et permettant d'établir la politique π_i de chaque agent $\mathcal{A}g_i$.

L'un des objectifs de notre travail est de résoudre des problèmes de planification de taille importante (grand nombre de tâches et d'agents). L'algorithme de calcul des politiques que nous allons mettre en place doit par conséquent faire preuve d'efficacité. Ainsi, nous allons rechercher une approximation de la politique optimale et limiter la complexité de la planification. Nous veillerons de plus à restreindre les mises à jour du modèle, en particulier celles concernant la fonction de transition.

Nous commencerons par décrire le principe de notre algorithme dont nous proposerons deux versions : l'une centralisée et la seconde décentralisée. Nous nous intéresserons ensuite à la complexité temporelle et spatiale de ces deux versions. Enfin, nous discuterons de la qualité des solutions obtenues et traiterons en particulier des cas d'optimalité. Les expérimentations portant sur cet algorithme seront quant à elles décrites dans la partie suivante du document traitant de la validation de nos travaux.

8.1 Algorithme de révision des politiques

Considérons un agent $\mathcal{A}g_i$ devant exécuter une tâche t_i à partir d'un état s_i. Soit t_j la tâche la plus proche de t_i dans le graphe de la mission qui soit exécutée par $\mathcal{A}g_j$ tel que $j \neq i$[40]. Le calcul de la politique d'exécution de t_i, décrit au chapitre précédent, réalise un compromis entre l'utilité espérée de l'agent $\mathcal{A}g_i$ et le coût occasionné sur les autres membres du système, parmi lesquels $\mathcal{A}g_j$. L'utilité espérée de $\mathcal{A}g_i$, lorsqu'il exécute t_i à st à partir de s_i, suppose la connaissance des probabilités de transition de l'agent à partir de s_i. Comme nous l'avons présenté au chapitre 6, le calcul de ces probabilités requiert que les politiques d'exécution des tâches en amont de t_i dans le graphe de la mission, soient connues. Le calcul du coût occasionné sur $\mathcal{A}g_j$ nécessite, quant à lui, de connaître l'utilité espérée de $\mathcal{A}g_j$ lorsqu'il exécute t_j. Décider de la politique d'exécution d'une tâche t_i suppose donc que nous connaissions la politique des tâches en amont de t_i et que l'utilité espérée de l'exécution des tâches en aval ait été calculée. Ainsi, nous allons, dans cette section, présenter un algorithme organisant le calcul des politiques des agents et permettant, à tout instant, d'avoir à notre disposition les informations nécessaires au calcul en cours.

8.1.1 Principe de l'algorithme

Cet algorithme de calcul des politiques doit garantir, lors de la prise en compte d'un état s_i, que nous connaissions la fonction de transition à partir de cet état. Pour ce faire, nous allons supposer que chaque agent suit une politique initiale connue et fixée, tout comme nous l'avons déjà fait au chapitre 6. Nous allons ensuite chercher à améliorer l'ensemble des politiques initiales des agents de façon à obtenir un nouvel ensemble augmentant la mesure de performance du système.

A partir des politiques initialement fixées, la fonction de transition de chaque agent peut être déterminée. Une fois la modélisation du problème établie, nous allons parcourir l'espace des états de chaque agent et calculer une nouvelle politique en chaque état en utilisant l'équation 7.4 présentée précédemment.

Néanmoins, dès que la politique d'exécution d'une tâche t_i change, les fonctions de transition des agents sont modifiées. Les modifications de la politique de t_i affectent en effet l'exécution des tâches en aval de t_i dans le graphe de la mission. Les probabilités de transition associées à l'exécution de ces tâches doivent alors être révisées. Reprenons le graphe de la figure 6.1. A partir d'un ensemble de politiques initiales, nous sommes en mesure de déterminer les fonctions de transition individuelles des agents 1, 2 et 3. Si nous modifions ensuite la politique d'exécution de la tâche D, les probabilités sur les dates de fin de D changent et les fonctions de transition des agents 1 et 2 initialement calculées deviennent en partie inexactes. En effet, les probabilités

[40] Cette tâche correspond à la première tâche accessible à partir de t_i dans le graphe de la mission qui est exécutée par $\mathcal{A}g_j$.

de transition, lors de l'exécution de E, F et G doivent être re-calculées en tenant compte de la nouvelle politique pour l'exécution de D.

Afin de garantir la continuité de la validité des fonctions de transition calculées à partir des politiques initiales, nous allons procéder à un ordonnancement des révisions des politiques des agents. Celui-ci nous permettra également de garantir la connaissance du coût occasionné lorsqu'un agent décidera d'exécuter une tâche t_i à st. L'algorithme de révision des politiques que nous allons proposer procède, pour ce faire, à un parcours du graphe de la mission des feuilles jusqu'aux racines. Les politiques d'exécution des feuilles du graphe sont donc tout d'abord déterminées. Soit t_i une de ces tâches et $\mathcal{A}g_i$ l'agent devant l'exécuter. Nous considérons alors tous les états à partir desquels t_i peut être exécutée, c'est-à-dire les états de succès et d'échec partiel associés à t_{i-1}, où t_{i-1} correspond à la tâche devant être exécutée avant t_i par $\mathcal{A}g_i$. En effet, la politique à partir d'un état associé à une tâche t_{i-1} exécutée par $\mathcal{A}g_i$, a trait à l'exécution de la tâche suivante t_i par $\mathcal{A}g_i$ puisque tout état $[t_{i-1}, *, *, *]$ représente une situation où la tâche t_{i-1} a été réalisée et la tâche suivante t_i doit être exécutée. Nous envisageons donc tous les états d'échec partiels $[t_{i-1}, *, *, *]$ et de succès $[t_{i-1}, *, *]$ et nous déterminons une nouvelle politique pour chacun d'eux en utilisant l'équation 7.4. Les politiques des tâches en amont de t_i dans le graphe de la mission n'ayant pas été encore modifiées, les fonctions de transition demeurent valides et peuvent être utilisées sans risque d'erreur. La tâche t_i n'ayant pas de successeur, aucune perte d'utilité n'est provoquée sur les autres agents, aucune valeur de coût occasionné n'a donc besoin d'être connue. Toutes les données nécessaires à l'utilisation de l'équation 7.4 sont donc à notre disposition. Une fois ces états évalués, nous calculons ensuite les valeurs de coût occasionné $OC_{t_i}(\Delta t, r_{t_i})$ pour tout couple $[\Delta t, r_{t_i}]$ où Δt correspond à un retard possible de l'exécution de t_i, et r_{t_i} à une quantité de ressources pouvant être disponible avant l'exécution de t_i.

Lorsque toutes les tâches t_i qui sont des feuilles du graphe ont été considérées, nous envisageons alors leurs prédécesseurs t_j. Si les politiques de tous les successeurs de t_j ont été révisées, nous sommes capable d'établir le coût occasionné provoqué par t_j sur les autres agents. La politique de cette tâche peut donc être révisée. Nous considérons alors tous les états d'échecs partiels $[t_{j-1}, *, *, *]$ et de succès $[t_{j-1}, *, *]$ associés à la tâche t_{j-1} et nous déterminons une nouvelle politique d'exécution de t_j pour chacun d'eux. Les politiques d'exécution des tâches en amont de t_j étant inchangées, la fonction de transition initialement calculée reste valide pour l'exécution de t_j. Les politiques d'exécution des tâches en aval de t_j ayant été révisées et par conséquent leur coût occasionné ayant été calculé, nous sommes en mesure de déterminer la perte en utilité provoquée par l'exécution de t_j à une date st. L'équation 7.4 peut alors être utilisée sans difficulté.

Le parcours de graphe réalisé par cette méthode de révision des politiques est très proche de ceux décrits aux chapitres 5 et 6. Nous commençons cependant par les feuilles du graphe, au lieu des tâches racines, et procédons par retour arrière. Chaque fois qu'une tâche est considérée, sa

politique d'exécution est révisée. Nous faisons donc évoluer simultanément les politiques de tous les agents. L'ordre dans lequel sont considérées les tâches garantit, à tout instant, la cohérence de la fonction de transition et la connaissance des valeurs de coût occasionné nécessaires à la détermination de la meilleure action à exécuter. Une fois tout le graphe considéré, la politique d'exécution de chaque tâche a été révisée. Nous obtenons ainsi un nouvel ensemble de politiques différent de l'ensemble des politiques initialement fixé[41].

8.1.2 Révision centralisée des politiques

Le principe de cet algorithme de révision des politiques peut être mis en œuvre de manière centralisée ou décentralisée. La version centralisée de l'algorithme consiste en l'amélioration des politiques des agents par une unité centrale. Cette dernière considère les tâches des agents une par une et améliore leur politique d'exécution.

La révision centralisée des politiques nécessite que la fonction de transition de chaque agent ait été préalablement définie et, de manière plus générale, que la modélisation du problème sous forme d'un OC-DEC-MDP soit achevée. Une fois ces données déterminées, le graphe de la mission est parcouru niveau par niveau, en partant des feuilles. Pour chaque tâche t_i d'un niveau donné, nous considérons l'ensemble des états s_i à partir desquels t_i peut être exécutée. L'utilité $V(s_i)$ ainsi que la politique $\pi_i(s_i)$ de chaque état s_i sont alors calculées en utilisant les équations 7.1 et 7.4. L'algorithme 7 détaille le principe de cet algorithme.

Pour chaque tâche t_i, nous envisageons tout d'abord l'ensemble des états d'échec partiel associés à t_{i-1}[42] puis, les états de succès associés à t_{i-1} sont considérés. En effet, les états d'échec partiel associés à t_{i-1} correspondent à des situations où la tâche t_{i-1} a été exécutée avec succès, l'agent tente alors d'exécuter sa prochaine tâche t_i et échoue partiellement. Les états d'échec partiel associés à t_{i-1} sont donc accessibles à partir des états de succès de t_{i-1}. De ce fait, il est nécessaire de connaître leur valeur (i.e. utilité) afin de déterminer l'utilité espérée à partir d'un état de succès associé à t_{i-1}. Leur évaluation doit par conséquent précéder celles des états de succès.

Pour des raisons similaires, il est également important d'évaluer les états d'échec partiel en ordre décroissant des dates de fin d'échec. Considérons deux états d'échec partiel $s_i = [t_{i-1}, [*, et], et(I')_{t_{i-1}}, r]$ et $s'_i = [t_{i-1}, [*, et'], et(I')_{t_{i-1}}, r']$. Si $et' > et$ et $r > r'$ alors, l'agent est en mesure de passer par s_i puis s'_i s'il échoue deux fois ou plus l'exécution de t_i. L'utilité espérée à partir de s_i dépend donc de la valeur de l'état s'_i qui doit être déterminée en premier.

La figure 8.1 illustre le fonctionnement de cet algorithme centralisé sur une partie du graphe de la mission déjà présenté sur les figures 6.1 et 4.4. A chaque étape d'évaluation des états et de

[41] Nous montrerons par la suite que dans certains cas particuliers ces deux ensembles peuvent toutefois être identiques.

[42] t_{i-1} correspond à la tâche exécutée par $\mathcal{A}g_i$ avant qu'il exécute t_i.

Algorithme 7 : Algorithme de révision centralisé

Entrées : un OC-DEC-MDP

$\pi =< \pi_1, \cdots \pi_n >$ l'ensemble des politiques initiales des agents

$Niv \leftarrow$ feuilles du graphe de la mission// `tâches du niveau courant`

$NivSuiv \leftarrow \emptyset$ // `tâches à considérer au niveau suivant`

répéter

  pour tous les $t_i \in Niv$ **faire**

    si *la politique de les successeurs de* t_i *a été révisée* **alors**

      // `Calcul de la politique d'exécution de` t_i

      Calculer la valeur de l'état d'échec total : $[failure_{t_{i-1}}, *, *]$

      pour tous les *st de* UB_{t_i} *à* LB_{t_i} **faire**

        pour tous les *taux de ressources* r_{t_i} *disponibles après un échec patiel de* t_i **faire**

          Calculer la valeur et la politique des états d'échec partiels : $[t_{i-1}, [st, st+1], et(I')_{t_{i-1}}, r_{t_i}]$

        fin

        pour tous les *taux de ressource* r_{t_i} *disponibles avant l'exécution de* t_i **faire**

          pour tous les δ_c^i *de* t_i **faire**

            Calculer la valeur et la politique des états de succès $[t_{i-1}, [st, st+\delta_c^i], r_{t_i}]$

          fin

          Calculer $V_{t_i}^{\Delta t, r_{t_i}}$ où $\Delta t = st - LB_{t_i}$

        fin

      fin

      pour tous les $V_{t_i}^{\Delta t, r_{t_i}}$ *calculées précédemment* **faire**

        Calculer $OC(\Delta t, r_{t_i}) = V_{t_i}^{0, r_{t_i}} - V_{t_i}^{\Delta t, r_{t_i}}$

      fin

      $NivSuiv \leftarrow NivSuiv \cup Pred(t_i)$

    sinon

      // `les successeurs n'ont pas tous été considérés`

      $NivSuiv \leftarrow NivSuiv \cup \{t_i\}$

    fin

  fin

  $Niv \leftarrow NivSuiv$

  $NivSuiv \leftarrow \emptyset$

jusqu'à $Niv = \emptyset$;

Sorties : Une nouvelle politique jointe π

calcul de la politique, nous indiquons sur la figure l'ensemble des états considérés. Pour chaque étape de calcul du coût occasionné, nous mentionnons les valeurs déterminées.

Le dernier niveau du graphe est tout d'abord considéré et la politique de la tâche G est alors calculée. Pour ce faire, tous les états associés à la tâche exécutée avant G par l'agent 2, c'est-à-dire F, sont pris en compte. Nous évaluons les états d'échecs partiel par ordre décroissant de la date de fin d'échec, en utilisant l'équation 7.1. Nous calculons également pour chacun de ces états la politique d'exécution de G à l'aide de l'équation 7.4. Les états de succès sont ensuite évalués et leur politique est déterminée. Enfin, nous déterminons les valeurs de coût occasionné de la tâche G. Une fois ces calculs réalisés, nous passons au niveau suivant du graphe. Nous évaluons alors tous les états associés à la tâche C afin de calculer la politique d'exécution de E. Nous utilisons pour ce faire les valeurs de coût occasionné sur G. Nous calculons ensuite les valeurs du coût occasionné associées à E et passons à la tâche suivante F. Lorsque la politique d'exécution de F est déterminée et que les valeurs de coût occasionné de la tâche sont connues, nous passons au niveau suivant, et ainsi de suite ...

Remarquons que les états associés à la dernière tâche de chaque agent ne sont pas pris en compte. En effet, ces états correspondent à des états terminaux à partir desquels aucune politique ne doit être déterminée. La valeur de ces états est aisément calculable puisqu'elle est égale à la récompense de la tâche à laquelle est associé l'état. Prenons l'exemple de l'état de succès $[F, [8, 10], 3]$ associé à la tâche F, sa valeur est égale à la somme du gain immédiat de l'agent (récompense de F) et de l'utilité espérée qui est nulle étant donné qu'aucune tâche ne sera ensuite exécutée par l'agent 2. Ainsi, il n'est pas nécessaire que notre algorithme considère ces états finaux.

Cette version centralisée de l'algorithme permet donc de faire évoluer les politiques des agents en améliorant à chaque étape la politique d'exécution des états associés à une tâche t_i. Quand une tâche t_i est prise en compte, toutes les tâches en aval de t_i dans le graphe de la mission ont déjà été considérées. L'utilité espérée de l'agent $\mathcal{A}g_i$ exécutant t_i, ainsi que les valeurs de coût occasionné, sont de ce fait calculées à partir des nouvelles politiques d'exécution. La figure 8.2 présente l'état des politiques lorsque t_i est considérée. Cette figure illustre le fait que l'algorithme améliore la politique de t_i en considérant les modifications déjà réalisées sur les politiques des tâches en aval. Lorsque le niveau du graphe de la mission contenant les tâches D et E est considéré, nous pouvons affirmer que les politiques d'exécution des tâches F, G et H ont été révisées. Le coût occasionné sur ces tâches est donc connu. Les tâches A, B et C en amont de D et E n'ont quant à elles pas encore été considérées. Les probabilités sur leurs dates de début et de fin, établies à partir des politiques initiales, sont donc toujours valides. Les fonctions de transition initialement calculées des agents 2 et 3 lors de l'exécution des tâches D et E demeurent par conséquent également valides.

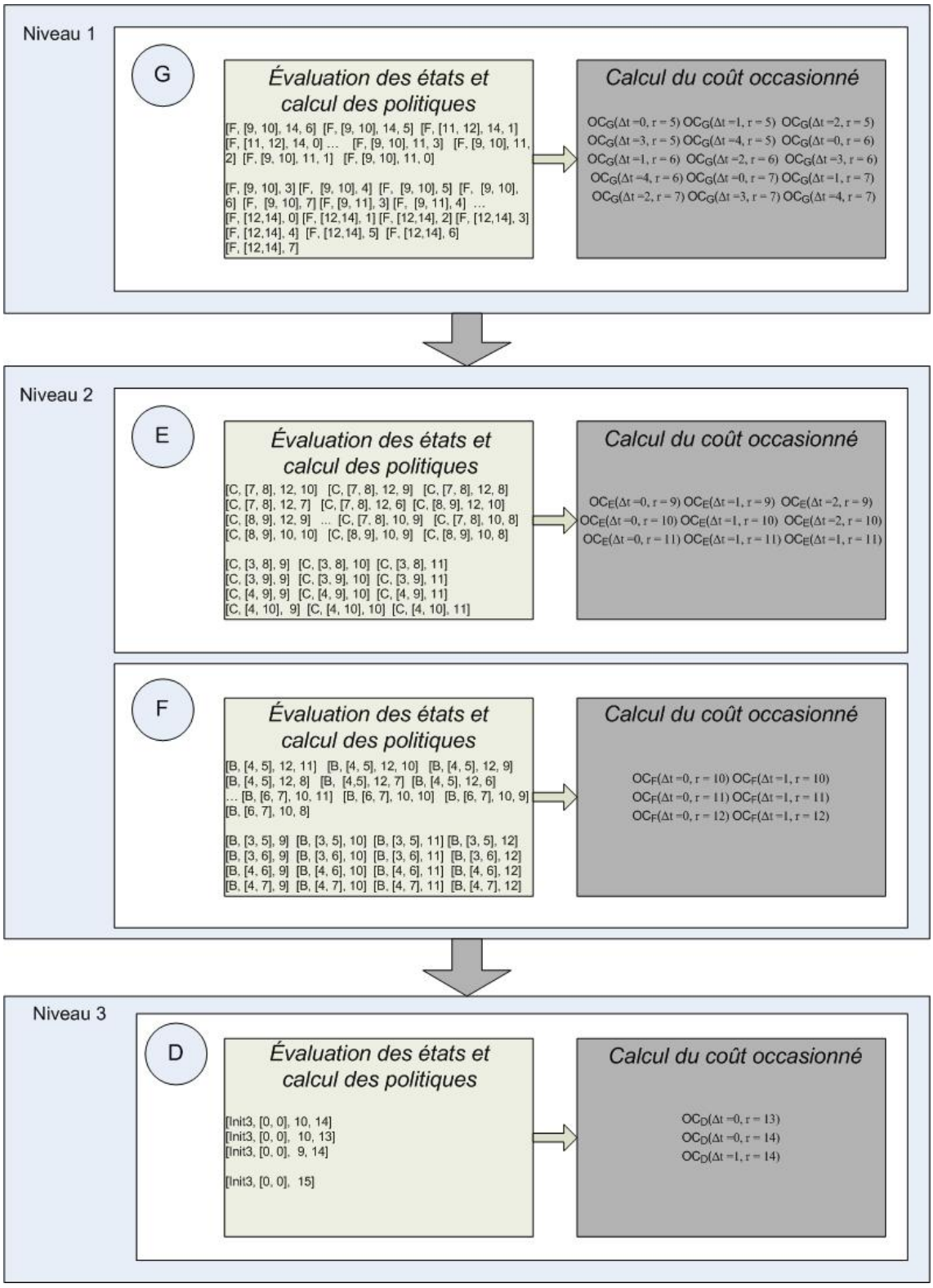


FIG. 8.1 – Trace d'exécution partielle de l'algorithme centralisé

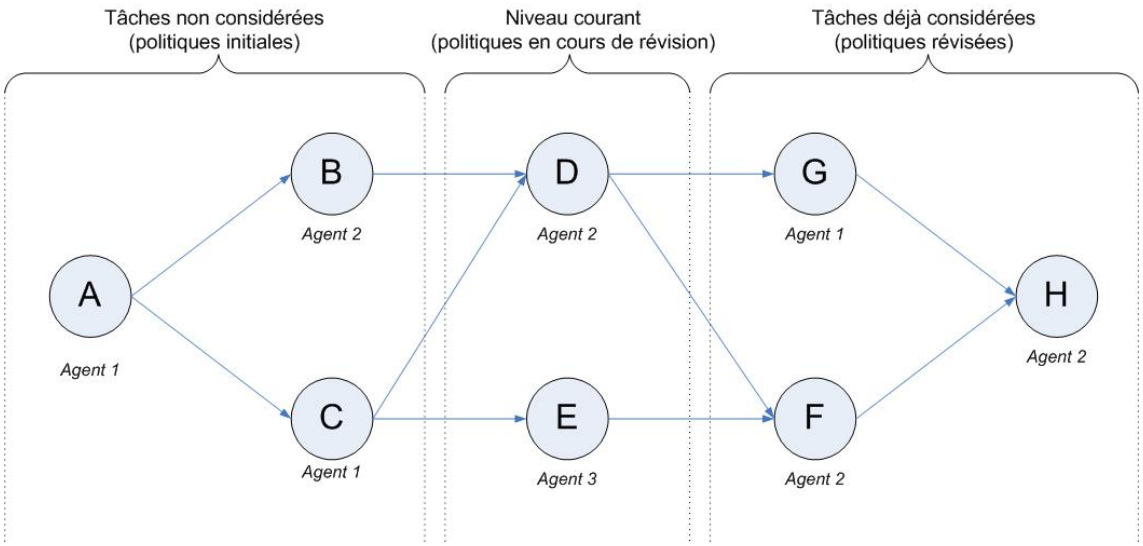


FIG. 8.2 – États des politiques lors de l'évaluation centralisée d'un niveau du graphe

L'utilisation de cet algorithme dans le cadre de la planification hors-ligne de missions multi-robots[43] nécessite l'existence d'une entité centrale réalisant le calcul des politiques puis, communiquant à chaque agent (robot) sa propre politique. Pour ce qui est des robots explorateurs sur Mars, cette phase de planification peut être réalisée par une machine sur Terre, chaque politique étant ensuite communiquée par satellite à l'agent concerné. Il est également possible d'envoyer le graphe de la mission à un agent planificateur sur Mars (satellite en orbite, station sur Mars, ou robot) ayant les capacités nécessaires pour réaliser la planification. Cet agent est ensuite chargé d'envoyer les politiques qu'il a calculées aux robots.

Dans tous les cas, la phase d'exécution des tâches consiste en l'application, par chaque robot, de sa fonction politique. Pour cela, un robot n'a besoin de connaître que sa propre politique. Les capacités mémoire et de calcul qui lui sont nécessaires pour prendre une décision sont donc limitées. Le processus délibératif étant très simple, les robots font de plus preuve d'une grande réactivité. Enfin, il est important de mentionner que les agents n'ont aucun besoin de communiquer avec les autres agents pour appliquer leur politique. Le résultat d'une telle planification remplit donc les exigences imposées au chapitre 3 concernant l'autonomie, la réactivité et l'absence de communication durant l'exécution.

[43] Nous renvoyons le lecteur au chapitre 4 pour la présentation des applications possibles de ce travail.

8.1.3 Révision décentralisée des politiques

Une version décentralisée de l'algorithme a également été mise en place afin de permettre à chaque agent de déterminer de lui même sa propre politique. Le principe de cet algorithme reste identique à ce qui a été précédemment décrit : nous supposons que chacun suit une politique initiale qui va évoluer par amélioration de la politique d'exécution de chaque tâche. Chaque agent va ainsi évaluer ses propres états et calculer sa nouvelle politique en conséquence.

Comme nous l'avons expliqué précédemment, la coordination des politiques nécessite la prise en compte du coût occasionné de chaque action. Pour qu'un agent puisse décider quelle action exécuter à partir d'un état s_i, il doit connaître le coût qu'il occasionne sur tout autre agent $\mathcal{A}g_j$. Ce coût occasionné dépend de l'utilité espérée de $\mathcal{A}g_j$ qui est calculée par ce dernier. Pour que $\mathcal{A}g_i$ puisse décider de l'exécution d'une tâche t_i, il doit donc avoir reçu des autres agents $\mathcal{A}g_j$ les valeurs du coût occasionné dont il a besoin. La version décentralisée de l'algorithme nécessite par conséquent que les agents puissent communiquer durant la phase de planification des tâches.

Afin de calculer sa propre politique, chaque agent doit connaître son propre MDP. L'agent parcourt alors, niveau par niveau, l'ensemble de son graphe de tâches[44]. Pour chaque tâche t_i, l'agent évalue tous les états à partir desquels t_i peut être exécutée (états associés à t_{i-1}) et il détermine sa politique d'action à partir de chacun d'eux. Pour ce faire, l'agent considère le coût occasionné par ses décisions sur les autres membres du système, et doit donc avoir reçu les valeurs de coût occasionné des tâches en aval de t_i dans le graphe de la mission.

Une fois la politique de la tâche t_i déterminée, l'agent calcule sa perte en utilité lorsque sa tâche t_i est retardée de Δt et qu'il possède r_{t_i} ressources. Ainsi, une valeur de coût occasionné est déterminée pour chaque couple $[\Delta t, r_{t_i}]$. Ces valeurs sont alors communiquées aux autres agents qui pourront les utiliser pour calculer leurs politiques. Une fois cette étape achevée, l'agent passe à la prochaine tâche à évaluer.

L'algorithme 8 décrit cette version décentralisée. Contrairement à la version centralisée, nous parcourons le graphe des tâches de chaque agent et non le graphe de la mission. Pour qu'un agent puisse calculer la politique à partir des états associés à t_i, il doit avoir reçu des autres agents les valeurs de coût occasionné qui lui sont nécessaires. Si ce n'est pas le cas, cela signifie que certaines tâches en aval de t_i n'ont pas encore été considérées. L'agent doit alors attendre de recevoir ces valeurs afin de poursuivre le calcul de sa politique.

La figure 8.3 présente le fonctionnement de cet algorithme sur une partie du graphe de mission représenté par les figures 6.1 et 4.4. L'ensemble des états évalués et des valeurs de coût occasionné à déterminer pour chaque tâche est identique à ce qui a été présenté pour la version décentralisée. Néanmoins, chaque agent évalue individuellement ses états. Ainsi, l'agent 2 commence à évaluer les états associés à la tâche F déterminant ainsi la politique d'exécution G. Puis, l'agent 2 calcule

[44] Rappelons que le graphe des tâches d'un agent ordonne les tâches que celui-ci doit exécuter (cf. chapitre 4).

Algorithme 8 : Algorithme de révision décentralisé

Entrées : Le MDP de l'agent $\mathcal{A}g_i$

$\pi =< \pi_1, \cdots \pi_n >$ l'ensemble des politiques initiales des agents

$Niv \leftarrow$ feuilles du graphe des tâches `// tâches du niveau courant`

$NivSuiv \leftarrow \emptyset$ `// tâches à considérer au niveau suivant`

répéter

  pour tous les $t_i \in Niv$ **faire**

    si *le coût occasionné de tous les successeurs de* t_i *a été reçu* **alors**

      `// Calcul de la politique d'exécution de` t_i

      Calculer la valeur de l'état d'échec total : $[failure_{t_{i-1}}, *, *]$

      pour tous les *st de* UB_{t_i} *à* LB_{t_i} **faire**

        pour tous les *taux de ressources* r_{t_i} *disponibles après un échec partiel de* t_i **faire**

          Calculer la valeur et la politique des états d'échec partiels : $[t_{i-1}, [st, st+1], et(I')_{t_{i-1}}, r_{t_i}]$

        fin

        pour tous les *taux de ressource* r_{t_i} *disponibles avant l'exécution de* t_{i-1} **faire**

          pour tous les δ_c^i *de* t_i **faire**

            Calculer la valeur et la politique des états de succès $[t_{i-1}, [st, st+\delta_c^i], r_{t_i}]$

          fin

          Calculer $V_{t_i}^{\Delta t, r_{t_i}}$ où $\Delta t = st - LB_{t_i}$

        fin

      fin

      pour tous les $V_{t_i}^{\Delta t, r_{t_i}}$ *calculées précédemment* **faire**

        Calculer et communiquer $OC(\Delta t, r_{t_i}) = V_{t_i}^{0, r_{t_i}} - V_{t_i}^{\Delta t, r_{t_i}}$

      fin

      $NivSuiv \leftarrow NivSuiv \cup Pred(t_i)$

    sinon

      `// les successeurs n'ont pas tous été considérés`

      $NivSuiv \leftarrow NivSuiv \cup \{t_i\}$

    fin

  fin

  $Niv \leftarrow NivSuiv$

  $NivSuiv \leftarrow \emptyset$

jusqu'à $Niv = \emptyset$;

Sorties : La politique π_i de l'agent $\mathcal{A}g_i$

les valeurs de coût occasionné de G qu'il envoie aux agents 1 et 3 en attente de ces valeurs. L'agent 1 peut alors entamer le calcul de la politique d'exécution de E et des valeurs de coût occasionné qui s'y rattachent. L'agent 2 considère, quant à lui, la tâche E et l'agent 3 est en attente. Une fois leurs calculs terminés, les agents 1 et 2 envoient respectivement, aux autres agents, les valeurs de coût occasionné des tâches E et F (envois matérialisés par des flèches sur la figure 8.3). Lorsque l'agent 3 reçoit ces valeurs, il peut entamer l'évaluation de la politique d'exécution de la tâche D, puisqu'il a alors toutes les informations nécessaires. Il considère ainsi les états associés à la tâche qu'il exécutera avant D. La tâche D étant la première tâche de l'agent 3, les états à considérer correspondent aux états associés à la tâche initiale fictive de l'agent 3 (« init 3 »).

Lorsqu'un agent détermine sa politique d'exécution pour une tâche t_i, il considère tous les états associés à sa tâche précédente t_{i-1}. A ce moment donné, nous pouvons affirmer que toutes les tâches en aval de t_i dans le graphe de la mission ont été considérées sinon, l'agent ne disposerait pas des valeurs de coût occasionné qui lui sont nécessaires. De plus, les tâches en amont de t_{i-1} dans le graphe des tâches n'ont pas encore été considérées puisque leur prise en compte nécessite que le coût occasionné sur t_{i-1} soit connu, ce qui sera le cas une fois la politique des états qui lui sont associés déterminée. La fonction de transition utilisée pour calculer l'utilité espérée à partir d'un état associé à t_{i-1} est donc toujours valide. La figure 8.4 décrit l'état des politiques au moment où la tâche t_i est prise en compte. Le graphe de la mission présenté en haut à gauche de la figure est décomposé en graphe de tâches. Lorsque l'agent 2 détermine la politique d'exécution de tâche D, l'agent 3 a également la possibilité de calculer sa politique pour E. Nous pouvons affirmer que l'agent 2 a, à ce moment donné, révisé sa politique d'exécution de F et H et que l'agent 1 a déjà considéré sa tâche G. Pendant que les agents 2 et 3 révisent leurs politiques, l'agent 1 est en attente des valeurs de coût occasionné des tâches D et E qui lui permettront de déterminer la politique d'exécution de la tâche C.

A la différence de la version précédente, la décentralisation permet de réviser simultanément la politique d'exécution de plusieurs tâches. En effet, les agents sont en mesure de calculer leur politique d'exécution en même temps, pourvu qu'ils aient à disposition les valeurs de coût occasionné nécessaires. Nous assistons donc à une révision simultanée des politiques des agents.

Appliquer cet algorithme dans le cadre de la planification multi-robot nécessite des capacités de calcul plus importantes de la part des robots, puisque la planification est réalisée « à bord ». Les robots doivent de plus être en mesure de s'envoyer des messages pendant cette phase de planification. L'utilisation de cet algorithme peut néanmoins être envisagée dans des applications comme l'exploration de Mars. Pour ce faire, les opérateurs sur Terre doivent envoyer à chaque robot son propre MDP. Si les robots ont la possibilité de se rassembler, ils peuvent alors consacrer une même partie de leur journée à la planification et échanger leurs valeurs de coût occasionné

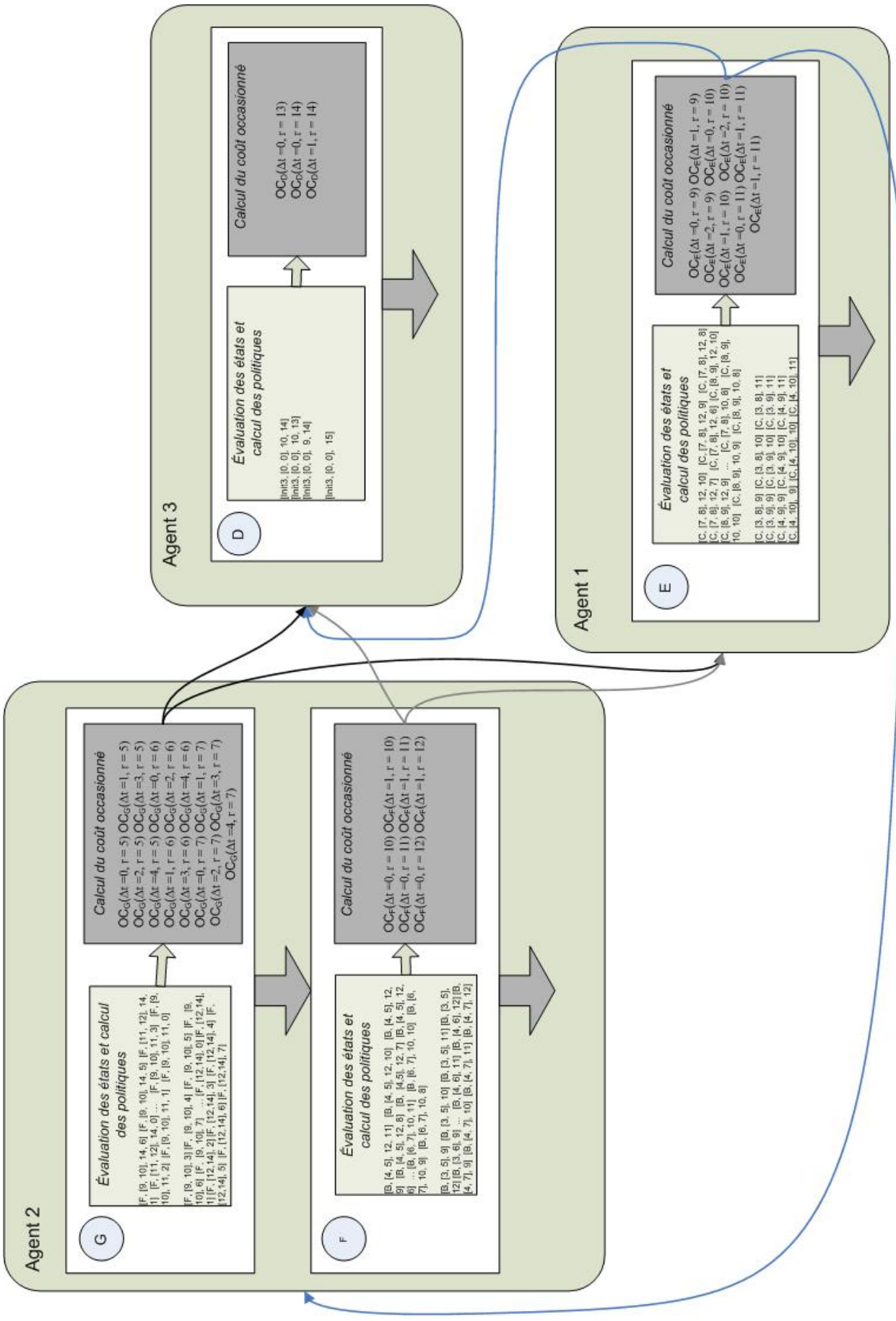


FIG. 8.3 – Trace d'exécution partielle de l'algorithme décentralisé

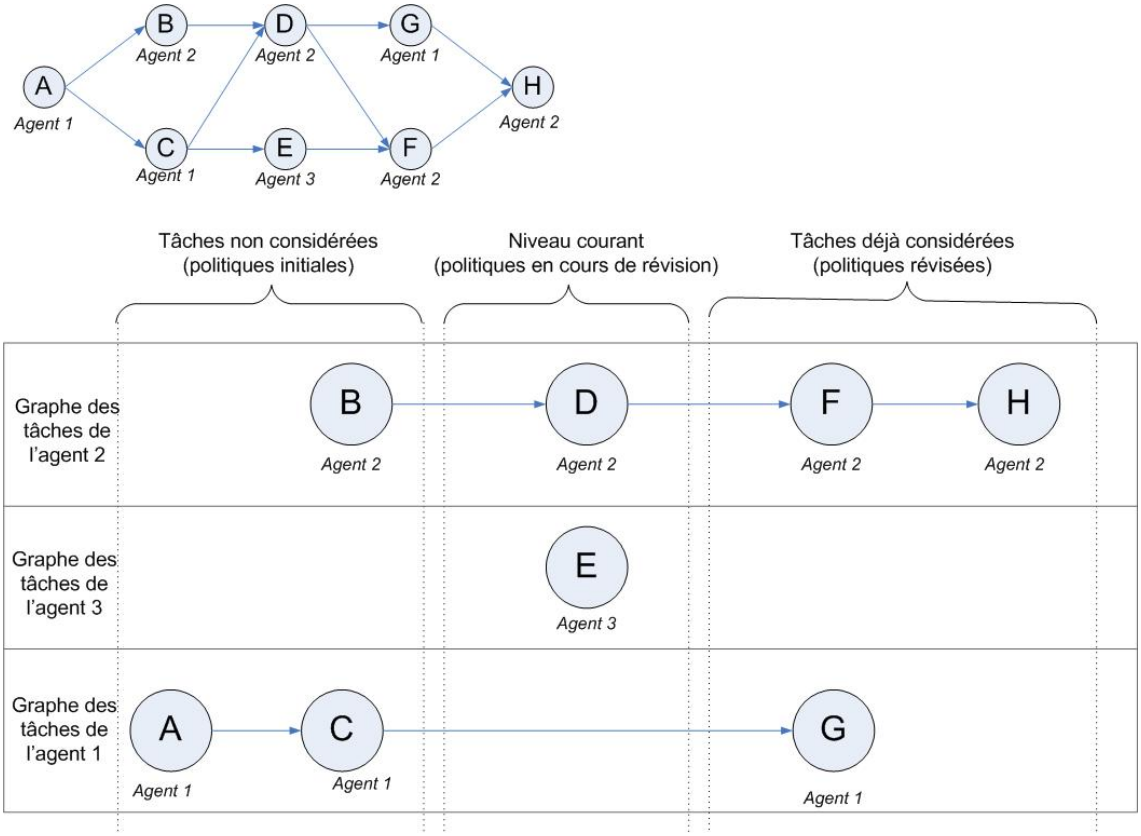


FIG. 8.4 – États des politiques lors de l'évaluation décentralisée des états d'une tâche

par communication directe (rendue possible grâce à la proximité). Si les robots sont très éloignés, la phase de planification doit avoir lieu pendant la période où le satellite est disponible, les robots échangeant alors leurs messages via celui-ci. Une fois cette planification hors-ligne réalisée, les robots n'ont plus besoin de communiquer. Ils peuvent donc, comme dans la version centralisée, agir de manière autonome et réactive sans avoir à échanger de messages. Notons qu'il est également envisageable d'envoyer à chaque robot le graphe de la mission. Chacun doit alors construire son propre MDP préalablement à l'exécution de l'algorithme qui vient d'être présenté.

8.2 Analyse de complexité

Dans la première partie de ce document réalisant l'état de l'art sur les processus décisionnels de Markov, nous avons mis en évidence les limitations des approches actuelles permettant de résoudre des DEC-MDPs. Nous avons décrit l'importante complexité des algorithmes existants, expliquant ainsi les restrictions portant sur la taille des problèmes jusqu'ici résolus. Afin de permettre la résolution de problèmes « moins académiques » et plus proches de ceux rencontrés dans les applications réelles, nous avons mis en avant la nécessité de développer des algorithmes de

résolution efficaces, i.e. de faible complexité. Nous allons à présent montrer que l'algorithme qui vient d'être présenté répond à ces exigences, que ce soit dans sa version centralisée ou décentralisée.

8.2.1 Dimension des données

La version centralisée, décrite par l'algorithme 7, parcourt l'ensemble des états des agents tout en calculant la valeur et la politique de chaque état. La version décentralisée parcourt, quant à elle, l'ensemble des états d'un agent et détermine pour chacun d'eux sa valeur et sa politique. Ainsi, la taille des espaces d'états des agents tient une place prépondérante dans la détermination de la complexité. Le nombre de valeurs de coût occasionné à communiquer et à mémoriser influence également de manière importante la complexité de notre algorithme. Nous allons donc tout d'abord nous intéresser aux dimensions de ces données pour ensuite établir la complexité des versions centralisées et décentralisées.

Taille de l'espace des états

Le nombre d'états succès $\#_{succ}$ d'un agent $\mathcal{A}g_i$ dépend du nombre de tâches $t_i \in \mathcal{T}_i$ qu'il doit exécuter, du nombre de dates de début et de dates de fin de chaque tâche et des quantités de ressources disponibles. Le nombre d'états succès au pire cas est donc défini par l'équation suivante :

$$\#_{succ} = \sum_{t_i \in \mathcal{T}_i} |ST_i| \times |ET_i| \times \#r_{t_i}$$

où $|ST_i|$ est le nombre de dates de début possibles de t_i, $|ET_i|$ est le nombre de dates de fin possibles de t_i et $\#r_{t_i}$ correspond au nombre de taux de ressources différents pouvant être disponibles avant l'exécution de t_i.

De façon similaire, le nombre d'états d'échecs partiels au pire cas est donné par :

$$\#_{EP} = \sum_{t_i \in \mathcal{T}_i} |ET_{i-1}| \times \#r_{t_i} \times |ST_i|$$

où $|ET_{i-1}|$ correspond au nombre de dates de fin de la tâche t_{i-1} exécutée juste avant t_i par $\mathcal{A}g_i$.

A chaque tâche est par ailleurs associé un état d'échec total. La taille de l'espace d'états $|\mathcal{S}_i|$ d'un agent $\mathcal{A}g_i$ est donc telle que :

$$\begin{aligned} |\mathcal{S}_i| &= \#_{succ} + \#_{EP} + |\mathcal{T}_i| \\ &= \sum_{t_i \in \mathcal{T}_i} \Big(|ST_i| \times |ET_i| \times \#r_{t_i} + |ET_{i-1}| \times \#r_{t_i} \times |ST_i| + 1 \Big) \end{aligned}$$

Valeurs de coût occasionné

Afin de déterminer la politique des agents, une valeur de coût occasionné est calculée et stockée pour chaque couple [retard, ressources] de chaque tâche. Dans la version décentralisée, ces valeurs sont également communiquées. Le nombre de retards possibles pour une tâche t_i est égal au nombre de dates de début possibles pour la tâche. Le nombre de valeurs de coût occasionné calculées par un agent est donc défini par :

$$\#_{OC_i} = \sum_{t_i \in \mathcal{T}_i} |ST_i| \times \#r_{t_i}$$

Remarquons dès à présent que $|\mathcal{S}_i| >> \#_{OC_i}$.

8.2.2 Complexité temporelle

A partir de la taille des données utilisées par notre algorithme, il nous est possible d'établir la complexité temporelle au pire cas des deux versions qui ont été présentées.

Corollaire 3 *La complexité temporelle au pire cas de l'algorithme de résolution centralisé d'un OC-DEC-MDP est polynomiale en $|\mathcal{SU}| \times |ST|$ où $|ST|$ désigne le nombre maximum de dates de début possibles pour une tâche et $\mathcal{SU}$ est égale à l'union des espaces d'états des agents. Nous avons donc $|\mathcal{SU}| = \sum_{\mathcal{A}g_i \in \mathcal{A}g} |\mathcal{S}_i| < |\mathcal{S}|$.*

Preuve : L'algorithme centralisé parcourt l'espace des états de chaque agent. Pour chaque état s_i, une valeur $V(s_i)$ et une politique $\pi_i(s_i)$ sont déterminées. Chacune de ces opérations a une complexité au pire cas en $O(|ST|)$. En effet, chaque action possible doit être envisagée. Une action consistant en l'exécution de la prochaine tâche à st, il existe au maximum $|ST|$ actions possibles à considérer.

Les lignes 7 à 9 de l'algorithme 7 considèrent tous les états d'échec partiel associés à t_i et calculent leurs valeurs et politiques, soit une complexité en $O(|ST| \times \#_{EP})$. Les lignes 10 à 16 considèrent tous les états de succès associés à une tâche t_i et calculent pour chacun d'eux leur valeur et leur politique d'où, une complexité en $O(|ST| \times \#_{succ})$. Les lignes 7 à 16 ont donc au final une complexité en $O(|ST| \times (\#_{succ} + \#_{EP}))$.

Les lignes 17 à 19 calculent les valeurs de coût occasionné associées à une tâche t_i. Le calcul des valeurs $V^{\Delta t, r_{t_i}}$ et $OC(\Delta t, r_{t_i})$ a une complexité en $O(1)$. Nous avons de plus montré qu'il existait, au pire cas, $|ST_i| \times \#r_{t_i}$ valeurs de coût occasionné pour chaque tâche. La complexité du calcul des valeurs de coût occasionné est donc en $O(|ST_i| \times \#r_{t_i})$

Nous pouvons de plus affirmer que :

$$|ST_i| \times \#r_{t_i} < |ST| \times (\#_{succ} + \#_{EP})$$

Les lignes 5 à 20 étant exécutées pour chaque tâche, la complexité temporelle de l'algorithme est de l'ordre de :

$$\sum_{t_k \in \mathcal{T}} |ST| \times (\#_{succ} + \#_{EP})$$

où :

$$\sum_{t_k \in \mathcal{T}} |ST| \times (\#_{succ} + \#_{EP}) = (\sum_{\mathcal{A}g_i \in \mathcal{A}g} |\mathcal{S}_i|) - |\mathcal{T}| \simeq |\mathcal{SU}|$$

La complexité temporelle de l'algorithme est alors en $O(|ST| \times |\mathcal{SU}|)$. Par conséquent l'algorithme est linéaire en $|ST| \times |\mathcal{SU}|$. □

Corollaire 4 *La complexité temporelle au pire cas de l'algorithme de résolution décentralisé d'un OC-DEC-MDP est polynomiale en $O(|ST| \times |\mathcal{SU}| + \#_{OC_i} \times K \times TM)$ où K désigne la taille d'un message et TM le temps de communication d'un bit d'information.*

Preuve : L'algorithme décentralisé permet à plusieurs agents d'évaluer leurs états en même temps. Néanmoins, toute évaluation des états associés à une tâche t_i nécessite que les tâches en aval de t_i aient déjà été considérées. Seuls les états associés aux tâches d'un même niveau peuvent donc être prises en compte simultanément.

Dans le pire cas, nous sommes en présence d'un graphe linéaire comme celui présenté sur la figure 8.5 et chaque niveau ne contient qu'une seule tâche. Nous ne pouvons donc considérer qu'une seule tâche à la fois et nous ne tirons pas parti de la décentralisation. Dans ce cas, la version décentralisée est donc équivalente, en temps d'exécution, à la version centralisée. Il est toutefois nécessaire de tenir compte du temps de communication des valeurs du coût occasionné. Cette dernière étant asynchrone, la complexité temporelle de l'envoi d'un message dépend de la taille du message K (nombre de bits) et du temps de transmission TM d'un bit. La complexité de l'envoi d'un message est alors en $O(K \times TM)$. Dans notre cas, un message correspond à une valeur représentée par un double, la taille du message reste donc limitée.

Nous pouvons en déduire que, dans le pire cas, la complexité temporelle de la version décentralisée est en $O(|ST| \times |\mathcal{SU}| + \#_{OC_i} \times K \times TM)$. □

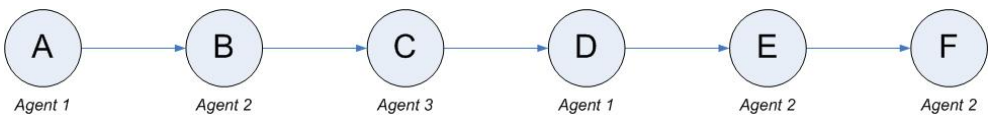


Fig. 8.5 – Exemple d'un graphe de mission linéaire

Les différences de complexité en moyenne entre les deux versions de l'algorithme dépendent du facteur de branchement du graphe (nombre de successeurs maximal pour une tâche). Dans le

meilleur des cas, chaque niveau du graphe contient une tâche pour chaque agent et la complexité de la version décentralisée de notre algorithme est en $O(|\mathcal{S}_i| \times |ST|)$.

8.2.3 Complexité en espace

Les différences de complexité entre la version centralisée de notre algorithme et la version décentralisée s'illustrent majoritairement par l'étude de la complexité spatiale.

Corollaire 5 *La complexité spatiale de l'algorithme de résolution centralisé d'un OC-DEC-MDP est polynomiale en fonction du nombre d'états des agents ($O(|\mathcal{SU}|)$).*

Preuve : Soit $\#n_{Max\_Res}$ le maximum du nombre de taux de ressources pouvant être disponibles avant l'exécution d'une tâche et $|ET|$ le nombre maximum de dates de fin d'une tâche. Pour chaque état s_i de chaque agent $\mathcal{A}g_i$, la politique en s_i doit être stockée (résultat de l'exécution de la ligne 12 de l'algorithme 7) à condition que s_i ne soit pas un état final. Il existe au maximum $\#finaux = |\mathcal{T}| + |\mathcal{A}g| \times \#n_{Max\_Res} \times |ST| \times |ET|$ états finaux (états d'échec total et états de succès associés à la dernière tâche de chaque agent). Ainsi $|\mathcal{SU}| - \#finaux$ valeurs sont à mémoriser. Nous devons par ailleurs garder en mémoire l'ensemble des valeurs de coût occasionné qui sont calculées, soit $\sum_{\mathcal{A}g_i \in \mathcal{A}g} \#_{OC_i}$ valeurs.

D'autres données sont également nécessaires à l'évaluation des états. Ainsi, pour chaque agent, le calcul de l'utilité $V(s_i)$ d'un état s_i associé à une tâche t_i nécessite que nous ayons à disposition les valeurs des états accessibles à partir de s_i. En raison de l'utilisation du principe d'optimalité de Bellman pour le calcul de $V(s_i)$, nous n'avons en effet pas besoin des valeurs de l'ensemble des états futurs de l'agent. Nous devons seulement stocker les valeurs des états d'échec partiel associés à t_i ainsi que les valeurs des états de succès de t_{i+1} (pour évaluer s_i) et de t_i (pour l'évaluation des états de la prochaine tâche à considérer). Soit un nombre de valeurs défini par :

$$\begin{aligned} &|\mathcal{A}g|(2(|ST| \times \#n_{Max\_Res} \times |ET|) + |ST| \times |ET| \times \#n_{Max\_Res}) \\ &\qquad = |\mathcal{A}g| \times (3(|ST| \times \#n_{Max\_Res} \times |ET|)) \end{aligned}$$

Enfin, chaque fois qu'une tâche t_i est considérée, nous avons temporairement besoin de garder en mémoire les valeurs $V^{\Delta t, r_{t_i}}$ nécessaires au calcul du coût occasionné (au pire $|ST| \times \#n_{Max\_Res}$ valeurs).

L'espace total $\mathcal{SP}_{cent}$ requis par l'exécution de l'algorithme est donc tel que :

$$\begin{aligned} \mathcal{SP}_{cent} &= |\mathcal{SU}| - \#finaux + \sum_{\mathcal{A}g_i \in \mathcal{A}g} \#_{OC_i} \\ &\quad + |\mathcal{A}g| \times (3(|ST| \times \#n_{Max\_Res} \times |ET|)) \\ &\quad + |ST| \times \#n_{Max\_Res} \\ &< 6|\mathcal{SU}| \end{aligned}$$

Nous en concluons que la complexité spatiale de l'algorithme est polynomiale en $|\mathcal{SU}|$. □

Corollaire 6 *L'algorithme centralisé de résolution d'un OC-DEC-MDP est PSPACE.*

Preuve : L'algorithme est polynomial en temps et en espace, il est donc PSPACE. □

Lorsque notre système ne comprend qu'un seul agent, les deux versions de l'algorithme consomment exactement le même espace mémoire puisque l'espace d'états d'un agent est égal à l'union des espace d'états de tous les agents. Dans le cas général où le système comprend plusieurs agents, l'espace mémoire consommé, pour chaque agent, par l'exécution de la version décentralisée de l'algorithme est moins important.

Corollaire 7 *La complexité spatiale de l'algorithme de résolution décentralisé d'un OC-DEC-MDP est polynomial en fonction du nombre d'états de l'agent et du nombre de valeurs du coût occasionné ($O(|\mathcal{S}_i| + \#_{OC})$).*

Preuve : La version décentralisée nécessite en effet de garder uniquement en mémoire la politique de l'agent considéré (et non celle de chaque agent). Le nombre d'états finaux d'un agent $\mathcal{A}g_i$ étant défini par : $\#finaux_i = |\mathcal{T}_i| + |\mathcal{A}g| \times (\#n_{Max\_Res} - 1) \times |ET|$, chaque agent $\mathcal{A}g_i$ doit garder en mémoire $|\mathcal{S}_i| - \#finaux_i$ valeurs.

En revanche, les valeurs de coût occasionné de tous les agents doivent être stockées ($\#n_{tasks} \times |ST| \times \#n_{Max\_Res}$ valeurs).

A chaque étape, nous ne mémorisons que les valeurs des états de l'agent nécessaires au calcul au lieu de stocker les valeurs des états de tous les agents. Soit : $3(|ST| \times \#n_{Max\_Res} \times |ET|)$ valeurs (le facteur $|\mathcal{A}g|$ disparaît).

L'espace $\mathcal{SP}_{decent}$ requis par l'exécution de la version décentralisée est donc défini par l'équation suivante :

$$\begin{aligned}
\mathcal{SP}_{decent} &= |\mathcal{S}_i| - \#finaux_i + \#n_{tasks} \times |ST| \times \#n_{Max_Res} \\
&\quad +3(|ST| \times \#n_{Max_Res} \times |ET|) \\
&\quad +|ST| \times \#n_{Max_Res} \\
&< 4|\mathcal{S}_i| + \#n_{tasks} \times |ST| \times \#n_{Max_Res} \\
&< 4|\mathcal{S}_i| + \#_{OC}
\end{aligned}$$

Nous en concluons que l'algorithme a une complexité spatiale en $O(|\mathcal{S}_i| + \#_{OC})$, il est donc polynomial. □

Corollaire 8 *L'algorithme décentralisé de résolution d'un OC-DEC-MDP est PSPACE.*

Preuve : L'algorithme est polynomial en temps et en espace, il est donc PSPACE. □

La quantité de mémoire utilisée par l'ensemble des agents est néanmoins supérieure à celle nécessaire pour exécuter la version centralisée. En effet, chacun doit stocker l'ensemble des valeurs du coût occasionné, d'où une redondance d'information. Cette quantité de mémoire totale est telle que :

$$\mathcal{SP}_{tot\_dec} = \mathcal{SP}_{cent} + (|\mathcal{A}g| - 1) \times \#_{OC}$$

8.2.4 Comparaison avec l'existant

L'étude de la complexité des versions centralisée et décentralisée de l'algorithme permet de supposer que des problèmes de taille importante pourront être résolus. En effet, contrairement aux approches existantes, nous proposons un algorithme de résolution polynomial en espace et en temps. Le tableau 8.1 met ces résultats en relation avec les complexités des algorithmes existants qui ont été présentés au chapitre 3.

Algorithme	OC-DEC-MDP centralisé	OC-DEC-MDP décentralisé	JESP	CSA	Bernstein et al.
Complexité	PSPACE	PSPACE	Exponentiel	Exponentiel	Exponentiel en espace Polynomial en temps

TAB. 8.1 – Comparaison de complexité avec les approches existantes

Le passage à une classe de complexité inférieure (polynomiale au lieu d'exponentielle) va nous permettre, comme nous le montrerons dans la partie suivante portant sur la validation du modèle, de surpasser les limitations qui étaient jusqu'à présent imposées sur la taille des problèmes pouvant être résolus.

8.3 A propos de l'optimalité

L'élaboration d'un algorithme polynomial a été possible grâce à une révision des exigences concernant l'optimalité de la solution. Le nombre de politiques jointes possibles étant tellement important, il est en effet difficile d'envisager trouver la solution optimale. Il est par conséquent préférable de se tourner vers la recherche d'une solution satisfaisante. Nous nous sommes par conséquent limitée à la recherche d'une approximation de la solution optimale. Nous avons ainsi déterminé un ensemble de politiques menant à une rationalité limitée au sens de H. Simon

(cf. chapitre 1). Une telle approche nous amène néanmoins à nous interroger sur la qualité des solutions obtenues par notre algorithme.

8.3.1 Qualité de la solution

A partir d'un même ensemble de politiques initiales, l'exécution de l'une ou l'autre des versions de notre algorithme mène à une même solution. En effet, ces deux versions utilisent exactement le même procédé et les mêmes données afin de déterminer la politique d'un agent à partir d'un état donné. Nous procédons, dans les deux cas, par améliorations successives des politiques initialement fixées en révisant une et une seule fois l'action à exécuter en chaque état. Les procédures de révision des politiques étant identiques, des résultats identiques sont obtenus.

En revanche, pour deux ensembles de politiques initiales différents, l'exécution de notre algorithme (quelle que soit la version) mène à deux solutions différentes. En effet, la fonction de transition de chaque agent, intervenant dans le calcul de l'utilité espérée, est déterminée à partir de ces politiques. La qualité de la solution est par conséquent dépendante des politiques initiales.

La méthode d'estimation du coût occasionné à laquelle nous avons recours lors du calcul de la politique dans un état s_i, influence également la qualité de la solution. Plus l'estimation du coût occasionné sera précise, mieux nous rendrons compte des effets d'une action et meilleure sera la décision. Nous détaillerons plus particulièrement dans la partie concernant la validation de notre modèle, l'influence de ces paramètres sur la qualité de la solution.

Nous pouvons néanmoins dès à présent remarquer que la procédure d'amélioration des politiques se trouve limitée par le fait que chaque état n'est considéré qu'une et une seule fois. Reprenons le graphe de la figure 6.1. Notre algorithme commence par améliorer le politique de la tâche G puis des tâches E et F. Bien que les modifications concernant la politique de la tâche E soient réalisées en prenant en compte leur influence sur la tâche G, il est possible, suite à la révision de la politique de E, que l'agent 2 ait tout intérêt à modifier à nouveau la politique de G puisque cette dernière a été établie en considérant que E serait exécutée suivant la politique initialement fixée de l'agent 1.

Notre algorithme permet donc d'améliorer successivement la politique en chaque état s_i en sélectionnant l'action a_i qui, à partir de s_i, fournit le meilleur compromis entre l'utilité espérée de l'agent et le coût occasionné sur les autres agents. Cette action maximise alors l'utilité espérée jointe des agents. En effet, nous pouvons démontrer le théorème suivant :

Corollaire 9 *L'action a_i maximisant l'équation 7.4 maximise également l'utilité espérée jointe des agents.*

Preuve : La preuve de ce théorème est détaillée en annexe A.2.1.

Ce théorème ne garantit néanmoins pas que la solution calculée par notre algorithme soit optimale.

8.3.2 Cas particuliers d'optimalité

L'étude des caractéristiques des problèmes de planification qui nous sont posés, nous a toutefois permis de répertorier un certain nombre de propriétés garantissant l'optimalité de la solution calculée par notre algorithme.

Les difficultés à décider quelle action exécuter à partir d'un état s_i émanent principalement des contraintes de ressources et des contraintes temporelles du problème. Rappelons que retarder l'exécution d'une tâche permet de diminuer les risques d'échec partiel, limitant ainsi les consommations de ressources qui y sont liées. En contrepartie, nous accentuons les risques de violer les contraintes temporelles des tâches à venir et nous augmentons la probabilité d'échec partiel des autres agents. Décider quand exécuter une tâche nécessite donc qu'un équilibre soit trouvé entre les différentes répercussions de ce choix sur l'utilité du système.

Néanmoins, lorsque les contraintes sur l'exécution des tâches sont relâchées, nous pouvons, dans certains cas, aisément identifer la politique optimale. Lorsque les ressources des agents sont illimitées par exemple, nous n'avons pas à nous préoccuper de l'impact des échecs partiels sur la disponibilité des ressources. Les conséquences de tels échecs sont alors inexistantes et il n'est pas besoin de tenir compte des ressources lors du calcul des politiques. Nous allons, dans cette section, identifier différentes propriétés conduisant à une simplification du choix des actions et permettant de déterminer aisément la politique optimale. Nous montrerons alors, que lorsque ces propriétés sont vérifiées, la solution retournée par notre algorithme est la politique optimale.

Pour cela, nous allons tout d'abord identifier deux politiques bien particulières : la politique EST et la politique LST. La première consiste à commencer toute exécution d'une tâche au plus tôt (Earliest Start Time). Pour chaque tâche, la première date de début possible est donc sélectionnée. Cette politique conduit généralement à de nombreux échecs partiels car plus nous essayons d'exécuter une tâche t_i tôt, plus la probabilité que les prédécesseurs de t_i n'aient pas terminé est forte. La seconde politique consiste à commencer l'exécution de chaque tâche le plus tard possible (Latest Start Time). Une telle politique conduit souvent rapidement à un échec total en raison de la violation des contraintes temporelles. En effet, plus l'exécution d'une tâche t_i débute tard, plus la probabilité de finir après la date de fin au plus tard est forte.

Lorsque le problème de planification qui nous est posé vérifie certaines propriétés, nous pouvons démontrer que l'une ou l'autre de ces politiques est optimale. Pour des raisons de clarté du document, les preuves des théorèmes qui vont suivre seront détaillées en annexe.

Théorème 2 *Si les ressources des agents sont illimitées alors, la politique EST est une politique optimale jointe.*

Théorème 3 *Si aucune contrainte temporelle n'est imposée sur les dates de fin des tâches ($LET \simeq +\infty$) alors, la politique LST est une politique optimale jointe.*

Théorème 4 *Si les ressources sont illimitées et qu'aucune contrainte temporelle n'est imposée sur les dates de fin alors, toutes les politiques sont équivalentes et optimales.*

Nous pouvons d'autre part démontrer que la résolution par notre algorithme de ces différentes classes de problèmes, conduit à une solution optimale.

Théorème 5 *Si les ressources des agents sont illimitées alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.*

Théorème 6 *Si aucune contrainte temporelle n'est imposée sur les dates de fin des tâches ($LET \simeq +\infty$) alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.*

Théorème 7 *Si les ressources sont illimitées et qu'aucune contrainte temporelle n'est imposée sur les dates de fin alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.*

Théorème 8 *Supposons que la politique initialement fixée de chaque agent soit la politique EST. Si le graphe de la mission ne comprend que deux agents Ag_i et Ag_j et que toutes les contraintes de précédence sont orientées de Ag_i vers Ag_j alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.*

Nous appelons un tel graphe « graphe à sens unique ». La figure 8.6 en présente un exemple. Une généralisation de ce théorème sera présentée au chapitre suivant.

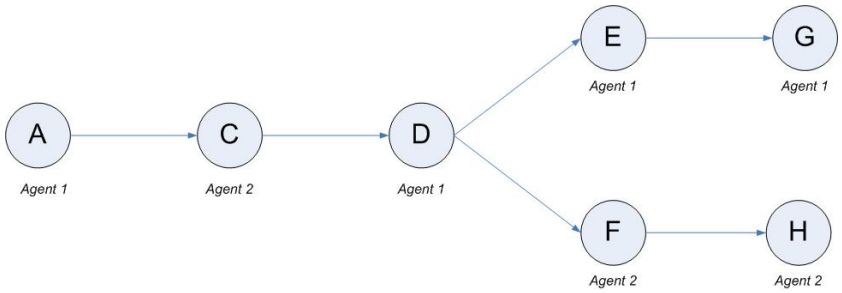


FIG. 8.6 – Exemple de graphe de mission à sens unique

Lorsque les ressources sont illimitées et/ou les contraintes temporelles sur les dates de fin d'exécution inexistantes et/ou le graphe de la mission est à sens unique, nous garantissons donc que notre approche calcule une politique optimale jointe. Lorsque le problème ne possède aucune

de ces propriétés, la politique optimale résulte d'un compromis entre la probabilité d'échouer partiellement, la consommation de ressources des échecs partiels, la probabilité d'échouer totalement par manque de ressources ou par violation des contraintes temporelles, et le gain obtenu par l'exécution des tâches. Tout changement dans la politique d'exécution d'une tâche risque donc de modifier cet équilibre et nécessite la révision des politiques des autres tâches.

Notre algorithme procède donc par améliorations successives de la politique d'exécution des tâches en ordonnant leur révision de sorte que les modifications déjà réalisées soient prises en compte. Comme nous l'avons déjà fait remarquer précédemment, le fait que chaque tâche ne soit considérée qu'une seule fois, limite la qualité de l'approximation. En effet, suite à la modification de la politique d'exécution d'une tâche t_i, seules les politiques d'exécution des tâches en amont de t_i sont révisées et peuvent donc tenir compte des répercussions de cette modification. Le fait que les politiques d'exécution des tâches en aval de t_i ne soient pas révisées, fait obstacle à l'obtention d'une situation d'équilibre et restreint la qualité de la solution. Il serait donc souhaitable de mettre en place un processus permettant de réitérer le parcours du graphe de façon à considérer chaque tâche plusieurs fois et améliorer la qualité de la solution.

Conclusion

Nous avons, dans ce chapitre, proposé un algorithme organisant le calcul des politiques des agents de sorte que les informations nécessaires à la détermination de la meilleure action à partir d'un état s_i, soient disponibles lorsque s_i est considéré. A partir d'un ensemble de politiques initialement fixé, cet algorithme procède à l'amélioration de la politique d'exécution de chaque tâche.

Deux versions de notre algorithme ont été décrites. La version centralisée permet à une unique entité de faire évoluer les politiques de tous les agents. La version décentralisée permet, quant à elle, l'évolution simultanée des politiques des agents. Ces derniers révisent en effet simultanément leur propre politique en utilisant les valeurs de coût occasionné qu'ils communiquent entre eux.

Après avoir détaillé ces deux versions de notre algorithme, leur complexité a été étudiée. Nous avons alors montré que l'une et l'autre était de complexité polynomiale. La mise en perspective de ce résultat avec les études de complexité des approches existantes, nous laisse présager que des problèmes de tailles importantes pourront être traités par les algorithmes issus de nos travaux. Nous reviendrons sur ce point au cours des expérimentations présentées dans la partie suivante de ce document.

Enfin, nous nous sommes intéressée à la qualité de la solution calculée par notre algorithme. Du fait de la complexité des problèmes que nous souhaitons résoudre, nous nous sommes précédemment orientée vers la recherche d'un approximation de la solution optimale. Mis à part les cas où ces problèmes possèdent des propriétés spécifiques que nous avons pu identifier, nous

ne pouvons pas garantir l'optimalité de la solution. Ce résultat est le prix à payer pour avoir la possibilité de résoudre des problèmes de taille conséquente.

L'étude de la qualité de la solution a par ailleurs mis en évidence le fait que notre algorithme ne permettait pas de réviser plusieurs fois la politique d'une même tâche, restreignant alors ses possibilités d'amélioration. Afin d'augmenter les performances de notre approche, nous allons, dans le prochain chapitre, corriger ces limitations.

Chapitre 9

Itération du processus d'approximation

Dans le chapitre précédent, nous avons présenté un algorithme permettant de faire évoluer au maximum les politiques des agents sans avoir à modifier la modélisation du problème, en particulier sans recalculer les fonctions de transition. Nous avons cependant mis en évidence les limitations de cet algorithme qui ne permet de réviser qu'une seule fois la politique de chaque tâche.

Nous allons à présent chercher à corriger ces limitations en développant une version itérative de l'algorithme précédemment décrit qui permettra de réviser plusieurs fois la politique d'exécution de chaque tâche, jusqu'à obtention d'un équilibre. Nous espérons ainsi augmenter la qualité des politiques obtenues. En contrepartie, il sera nécessaire, au cours de l'exécution de l'algorithme, de recalculer les fonctions de transition. Nous chercherons, néanmoins, à limiter la fréquence de ces calculs.

Nous commencerons par présenter le principe de l'algorithme itératif que nous avons mis en place. Deux versions de cet algorithme (centralisée et décentralisée) seront alors décrites. Nous discuterons ensuite de leur complexité temporelle et spatiale, ce qui nous amènera à aborder la question de la convergence. Enfin, nous nous intéresserons à la question de l'optimalité des politiques ainsi calculées.

9.1 Algorithme itératif d'amélioration des politiques

Afin d'améliorer la qualité des politiques calculées par notre algorithme, il est nécessaire de pouvoir considérer chaque tâche plusieurs fois. Ainsi, nous pourrons espérer atteindre une situation d'équilibre dans laquelle toute modification de la politique d'exécution d'une tâche conduirait à une solution de moins bonne qualité.

9.1.1 Principe du processus itératif

Afin d'obtenir un tel équilibre, nous avons mis en place une version itérative [Beynier et Mouaddib, 2006a, Beynier et Mouaddib, 2006b] de notre algorithme précédemment présenté. Ce dernier consiste en l'amélioration d'un ensemble de politiques initiales et termine lorsque chaque tâche a été considérée une fois, retournant alors un nouvel ensemble de politiques. Dans le but d'obtenir des solutions de plus grande qualité, nous proposons de re-exécuter cet algorithme en considérant, lors de cette nouvelle exécution, l'ensemble des politiques qui vient d'être calculé comme l'ensemble des politiques initiales. Ainsi, à l'itération N, l'ensemble des politiques initiales considéré correspond à l'ensemble résultant de la $N-1^{\text{ème}}$ itération. Le processus de révision des politiques est ainsi répété jusqu'à ce qu'aucun changement ne soit plus possible, c'est-à-dire que l'ensemble des politiques initiales et l'ensemble des politiques finales d'une même itération soient identiques.

Les fonctions de transition individuelles des agents étant établies à partir de l'ensemble des politiques initiales, il est nécessaire, à chaque itération, de les mettre à jour en considérant les politiques qui viennent d'être calculées. Le reste du modèle (espace d'états, d'actions, etc.) reste en revanche identique à chaque itération et ne requiert par conséquent aucune modification. L'ordonnancement de la révision des tâches réalisé à chaque itération limite au maximum les modifications du modèle. En effet, celui-ci est gardé valide le plus longtemps possible, ce qui permet de réaliser le maximum de modifications à partir d'une modélisation donnée. Si nous n'ordonnancions pas les tâches au cours d'une même itération, il serait nécessaire de recalculer les fonctions de transition avant d'avoir considéré toutes les tâches. La fréquence des révisions serait alors fortement accrue, diminuant ainsi l'efficacité de l'approche.

Considérons deux tâches t_i et t_j telles que t_j soit placée en amont de t_i dans le graphe de la mission. Itérer l'algorithme de révision des politiques permet de considérer plusieurs fois la politique d'exécution de ces tâches. A la première itération, t_i et t_j sont supposées être exécutées en suivant les politiques initialement fixées. La politique de t_i est alors révisée puis, t_j est considérée. La politique d'exécution de t_j est alors calculée en tenant compte de la révision de la politique de t_i. Grâce à l'algorithme itératif, une fois la politique de t_j modifiée, il est possible de modifier à nouveau la politique de t_i en fonction de la nouvelle politique d'exécution de t_j.

9.1.2 Formalisation de l'algorithme itératif

Deux versions de notre algorithme de révision des politiques ont été précédemment décrites. De la même façon, deux versions (centralisée et décentralisée) de l'algorithme itératif peuvent être envisagées.

Version centralisée

La première version décrite par l'algorithme 9 consiste à exécuter itérativement la version centralisée de notre algorithme de révision des politiques présenté au chapitre précédent. Au début de chaque itération, les fonctions de transition individuelles sont re-calculées à partir du nouvel ensemble de politiques initiales, en utilisant l'algorithme 5 de propagation des probabilités. L'algorithme d'amélioration centralisé (algorithme 7) des politiques est ensuite exécuté afin de déterminer le nouvel ensemble de politiques qui servira de politiques initiales à l'itération suivante. Ce processus est ainsi répété jusqu'à ce qu'aucune modification des politiques ne soit plus possible. Pour ce faire, le nombre de modifications des politiques est comptabilisé à chaque itération. Chaque fois qu'un état s_i est considéré la politique calculée est comparée avec la politique initialement fixée au début de l'itération pour s_i. La fonction $diff(\mathcal{A}g_k)$, à laquelle il est fait appel à la ligne 6 de l'algorithme 9, retourne alors le nombre de modifications de la politique de l'agent $\mathcal{A}g_k$ qui ont été comptabilisées.

Algorithme 9 : Algorithme itératif de révision centralisé

Entrées : un OC-DEC-MDP

$\pi =< \pi_1, \cdots \pi_n >$ l'ensemble des politiques initiales des agents

1 **répéter**

`/* Mise à jour du modèle */`

2 Calculer les fonctions de transition individuelles à partir de π

`/* révision centralisée des politiques */`

3 $\pi' \leftarrow$ résultat de l'exécution de l'algorithme de révision centralisé des politiques

`/* nombre de changements dans les politiques des agents */`

4 nbChangements $\leftarrow 0$

5 **pour tous les** $\mathcal{A}g_k \in \mathcal{A}g$ **faire**

6 nbChangements += diff($\mathcal{A}g_k$)

7 **fin**

8 $\pi \leftarrow \pi'$

9 **jusqu'à** *nbChangements == 0* ;

Sorties : Une nouvelle politique jointe π

Version décentralisée

La version décentralisée de l'algorithme itératif fonctionne sur le même principe, la révision des politiques étant cependant réalisée par chaque agent. Cette version est présentée par l'algorithme 10.

A chaque exécution de l'algorithme de révision des politiques, chaque agent comptabilise le

Algorithme 10 : Algorithme itératif de révision décentralisé

Entrées : Le MDP de l'agent $\mathcal{A}g_i$

$\pi =< \pi_1, \cdots \pi_n >$ l'ensemble des politiques initiales des agents

répéter

`/* Mise à jour du modèle */`

Calculer la fonction de transition individuelle de l'agent $\mathcal{A}g_i$ à partir de π

`/* révision décentralisée de la politique de l'agent */`

$\pi'_i \leftarrow$ résultat de l'exécution de révision décentralisé des politiques

`/* Envoi de` π'_i `aux autres agents */`

pour tous les $\mathcal{A}g_j \in \mathcal{A}g, i \neq j$ **faire**

Envoyer π'_i à $\mathcal{A}g_j$

fin

`/* calcul du nombre de changements dans les politiques des agents */`

nbChangements $\leftarrow 0$

pour tous les $\mathcal{A}g_k \in \mathcal{A}g$ **faire**

nbChangements += diff($\mathcal{A}g_k$)

$\pi_k \leftarrow \pi'_k$

fin

jusqu'à $nbChangements == 0$;

Sorties : La politique π_i de l'agent $\mathcal{A}g_i$

nombre de modifications qu'il a effectué sur sa politique. Si à la fin d'une itération, aucun agent n'a réalisé de changement, un équilibre est obtenu et l'algorithme s'arrête. Pour que chaque agent déduise qu'un équilibre est atteint, il est nécessaire que chacun sache si les autres agents ont ou non modifié leurs politiques. Ces informations doivent par conséquent être communiquées une fois les politiques révisées. La fonction $diff(\mathcal{A}g_k)$ renvoie ainsi le nombre de modifications que l'agent $\mathcal{A}g_k$ a réalisé sur sa politique, cette valeur étant envoyée par $\mathcal{A}g_k$ à tous les autres membres du système.

Le calcul de la nouvelle fonction de transition nécessite par ailleurs, que les politiques des agents soient connues de tous. Ce calcul peut être réalisé par une unité centrale ayant connaissance de toutes les politiques ou bien par les agents eux-mêmes. Dans ce dernier cas, il est nécessaire que chacun communique, à la fin de chaque itération, sa politique à tous les autres membres du système. Pour limiter ces communications, il est possible que les agents n'envoient que la politique des états qui ont été modifiés. Dans tous les cas, la fréquence de communication de ces informations est bien moins importante que dans le cadre de l'exécution décentralisée d'algorithmes comme celui proposé par Chadès et al. [Chadès *et al.*, 2002]. En effet, nous révisons

entièrement les politiques des agents avant de les communiquer. Dans le cas de l'algorithme de co-évolution proposé par Chadès et al. [Chadès *et al.*, 2002] ou de JESP [Nair *et al.*, 2003], seule une partie des politiques des agents est révisée à chaque itération, c'est-à-dire avant chaque phase de communication des politiques.

9.2 Complexité et convergence

La complexité de ces deux versions de l'algorithme itératif repose sur la complexité d'une itération et sur le nombre d'itérations réalisées. En nous appuyant sur l'étude de complexité qui a été précédemment menée concernant l'algorithme de révision des politiques, nous allons nous intéresser à la complexité temporelle et spatiale de l'algorithme itératif. Afin d'établir une borne supérieure sur le nombre d'itérations de cet algorithme, nous serons ensuite amenés à discuter de sa convergence.

9.2.1 Complexité en temps

Soit $\mathcal{IN}$ le nombre d'itérations au pire cas de l'algorithme itératif. La complexité temporelle de ce dernier dépend alors de $\mathcal{IN}$, ainsi que de la complexité temporelle d'une itération.

A chaque itération, le système de transition est mis à jour et les politiques d'exécution des tâches sont révisées en utilisant l'une ou l'autre des versions de l'algorithme présentées au chapitre précédent. Le calcul des probabilités de transition est réalisé à l'aide de l'algorithme 5 dont la complexité temporelle est, comme nous l'avons démontré au chapitre 6, polynomiale en espace d'états. A partir de ces informations, nous pouvons donc déduire la complexité des versions centralisées et décentralisées de l'algorithme itératif.

Corollaire 10 *La complexité temporelle au pire cas de l'algorithme itératif centralisé est polynomiale en* $\mathcal{IN} \times |\mathcal{SU}| \times |ST|$.

Preuve : La complexité temporelle de l'algorithme de révision centralisé est en $O(|\mathcal{SU}| \times |ST|)$. La complexité de l'algorithme de calcul des probabilités de transition étant en $O(|\mathcal{SU}|)$, la complexité d'une itération est par conséquent en $O(|\mathcal{SU}| \times |ST|)$. Ceci nous conduit à une complexité temporelle globale en $O(\mathcal{IN} \times |\mathcal{SU}| \times |ST|)$. □

Corollaire 11 *La complexité temporelle au pire cas de l'algorithme itératif décentralisé est en* $O(|ST| \times |\mathcal{SU}| + (\#_{OC_i} + |\mathcal{S}_i| + 1) \times K \times TM)$.

Preuve : A chaque itération, l'agent $\mathcal{A}g_i$ révise ses probabilités de transition à l'aide de l'algorithme 5 dont la complexité est polynomiale en $|\mathcal{SU}|$. L'algorithme de révision décentralisé, dont la complexité est en $O(|ST| \times |\mathcal{SU}| + \#_{OC_i} \times K \times TM)$, est ensuite exécuté. Enfin,

l'agent envoie aux autres membres du système, sa politique et le nombre de modifications qu'il a réalisées, soit $|\mathcal{S}_i| + 1$ valeurs chacune stockée sur un double. La complexité temporelle d'une itération est par conséquent en $O(|ST| \times |\mathcal{SU}| + (\#_{OC_i} + |\mathcal{S}_i| + 1) \times K \times TM)$, soit une complexité temporelle globale en $O(|ST| \times |\mathcal{SU}| + (\#_{OC_i} + |\mathcal{S}_i| + 1) \times K \times TM)$. □

La complexité en moyenne de l'algorithme itératif dépend du nombre d'itérations et de la complexité en moyenne de l'algorithme de révision des politiques. Cette dernière dépend, dans le cas d'une exécution décentralisée, du facteur de branchement du graphe de la mission. Le nombre d'itérations est lié pour sa part aux politiques initialement fixées et utilisées lors de la première itération. Il est également influencé par la méthode d'estimation du coût occasionné à laquelle il est fait appel afin de déterminer la politique à partir d'un état. Nous reviendrons plus en détail sur ce point lors de la présentation des expérimentations qui ont été mises en œuvre pour valider notre approche.

9.2.2 Complexité en espace

Contrairement à la complexité temporelle, la complexité en espace de l'algorithme itératif n'est pas influencée par le nombre d'itérations. A chaque itération, l'espace mémoire peut en effet être ré-utilisé, il n'est par exemple pas nécessaire de conserver les valeurs de coût occasionné ni les politiques des itérations précédentes. L'espace mémoire consommé reste ainsi constant au fil de l'exécution de l'algorithme.

Corollaire 12 *La complexité spatiale de l'algorithme itératif centralisé est polynomiale en nombre d'états des agents.*

Preuve : L' algorithme de révision des probabilités de transition et l'algorithme de révision centralisé ont tous deux une complexité spatiale en $O(|\mathcal{SU}|)$. L'espace consommé par une itération étant identique à celui consommé par $\mathcal{IN}$ itérations, nous en déduisons que l'algorithme est polynomial en $|\mathcal{SU}|$. □

Corollaire 13 *La complexité spatiale de l'algorithme itératif décentralisé est polynomiale en nombre d'états des agents.*

Preuve : La complexité spatiale de l'algorithme de révision des probabilités de transition est polynomial en $|\mathcal{SU}|$ puisqu'il nécessite de connaître l'ensemble des politiques des agents. En revanche, l'algorithme de révision décentralisé a une complexité spatiale en $O(|\mathcal{S}_i| + \#_{OC})$. Étant donné que $|\mathcal{S}_i| + \#_{OC} << |\mathcal{SU}|$, nous déduisons que la complexité spatiale de l'algorithme itératif décentralisé est polynomiale en $|\mathcal{SU}|$.□

9.2.3 Convergence de l'algorithme

Étant donnée l'importance du nombre d'itérations sur la complexité temporelle de l'algorithme itératif, nous sommes naturellement amenée à nous interroger sur la convergence de ce processus itératif. Celle-ci permet en effet de garantir qu'une solution finale soit retournée par notre algorithme, en d'autres termes que le nombre d'itérations soit fini.

Hypothèses de convergence

Nous avons précédemment démontré (cf. corollaire 9) que la politique d'un agent $\mathcal{A}g_i$, à partir d'un quelconque état s_i, était déterminée de façon à maximiser l'utilité espérée du système. Nous pouvons alors nous interroger sur l'évolution de cette utilité lorsque plusieurs agents décident en même temps, et sans se consulter, de leurs politiques.

En effet, l'algorithme que nous avons présenté procède à la révision simultanée[45] des politiques d'exécution des tâches d'un même niveau du graphe de la mission. Bien que la politique de chaque agent soit calculée de sorte que l'utilité espérée jointe soit maximisée, nous pouvons nous demander si le fait que les politiques de plusieurs agents soient modifiées en même temps ne risque pas de mener à la dégradation de l'utilité espérée du système. Il est alors nécessaire d'étudier l'influence conjuguée des politiques révisées sur l'utilité. Si nous sommes en mesure de démontrer qu'un tel processus de révision des politiques ne peut conduire à une baisse de l'utilité espérée jointe, nous pourrons alors conclure à la convergence de l'algorithme itératif. En effet, l'utilité espérée du système étant bornée supérieurement, si chaque itération conduit à augmentation de l'utilité du système alors, l'algorithme converge.

Preuve de la convergence

Avant de procéder à l'étude de la convergence de l'algorithme itératif, nous tenons à clarifier la terminologie que nous allons employer. Une valeur de coût occasionné calculée « correctement » ou « exactement » désignera, dans nos propos, une valeur déterminée sans erreur par rapport à une méthode de calcul donnée. Ainsi, les valeurs établies sans tenir compte des ressources seront correctes ou exactes si les données utilisées pour effectuer ses calculs sont à jour et qu'aucune erreur de calcul n'a été réalisée. Comme nous l'avons remarqué précédemment, ces valeurs peuvent néanmoins ne pas rendre compte fidèlement de l'influence d'une action, elles seront alors dites « imprécises ». Ce manque de précision émane cependant de la méthode de calcul utilisée et non du calcul en lui même. Nous avons décrit en section 7.3 une méthode permettant, lorsque les

[45] Dans la version centralisée de l'algorithme, les tâches d'un même niveau sont considérées successivement. Néanmoins, nous ne tirons pas parti de cette séquentialité. Nous sommes donc dans une situation équivalente à la révision simultanée des politiques d'exécution des tâches d'un même niveau.

valeurs sont correctement déterminées, d'obtenir une mesure précise. Pour ce faire, nous devons tenir compte des probabilités de transition ainsi que des ressources disponibles.

Sous certaines conditions, nous avons pu prouver que l'utilité espérée du système augmente lorsque les politiques d'exécution des tâches d'un même niveau sont modifiées simultanément. En effet, si le coût occasionné est correctement estimé et chaque agent suit initialement la politique EST alors, l'utilité espérée du système augmente à chaque itération. Afin de ne pas surcharger de façon conséquente ce document, nous ne détaillerons pas la preuve de la convergence sous de telles hypothèses. Si le coût occasionné est correctement estimé et que chaque agent suit initialement la politique EST, cette démonstration prouve que les politiques d'exécution des tâches appartenant à un même niveau L_k du graphe sont révisées de façon à ce que la décision de chaque agent maximise l'utilité du système quelles que soient les politiques d'exécution des autres tâches de L_k.

Exactitude et précision du coût occasionné

En raison des interactions entre les tâches, estimer le coût occasionné correctement peut néanmoins s'avérer difficile. Considérons une tâche t_i d'un niveau L_i. Les politiques d'exécution des tâches t_j ($j \neq i$) appartenant au même niveau L_i que t_i peuvent conditionner le coût occasionné par t_i. Elles influencent en effet les disponibilités de ressources lors de l'exécution des tâches en aval dans dans le graphe de la mission. Si les politiques des tâches t_j sont révisées, les ressources disponibles peuvent être modifiées et l'estimation du coût occasionné précédemment établie peut ainsi être faussée.

Reprenons le graphe de la figure 6.1. Les politiques d'exécution des tâches E et F sont révisées simultanément. Le coût occasionné par l'agent 1 sur l'agent 2 (coût occasionné sur G), et utilisé pour ces calculs, dépend des ressources disponibles avant l'exécution de G. Ces dernières sont quant à elles influencées par la politique d'exécution de la tâche F. Le coût occasionné sur G, utilisé pour déterminer la politique d'exécution de E, est calculé en supposant que F est exécutée suivant la politique initialement fixée pour l'agent 2. Si la politique d'exécution de F est modifiée, le coût occasionné devient alors inexact. Ceci est lié au fait que le coût occasionné sur G dépend indirectement de la politique d'exécution de F qui est modifiée simultanément à celle de E.

Considérons à présent le niveau constitué des tâches B et C. Nous sommes en mesure de garantir l'exactitude du coût occasionné par B (ou par C) sur l'agent 3 puisqu'aucune tâche n'est exécutée par celui-ci dans le niveau considéré. En revanche, le coût occasionné provoqué par l'agent 2 sur l'agent 1 dépend de la politique de C et peut être incorrectement estimé. Il en est de même pour le coût occasionné par l'agent 1 sur l'agent 2 qui dépend quant à lui de la politique d'exécution de B.

Les difficultés à déterminer de manière exacte les valeurs de coût occasionné sont dues à

l'influence des ressources sur cette mesure. Nous avons en effet précédemment préconisé l'utilisation d'une mesure de coût occasionné qui prenne en compte les ressources et estime ainsi le plus précisément possible l'impact d'une action. Lorsque la politique d'une tâche t_i d'un agent $\mathcal{A}g_i$ est modifiée, les ressources disponibles après l'exécution de t_i, et par conséquent avant toute exécution de la tâche suivante t_{i+1}, changent. Les valeurs de coût occasionné associées à t_{i+1} et précédemment déterminées deviennent alors inexactes et doivent être re-calculées. Pour ce faire, les probabilités sur les ressources disponibles avant l'exécution de t_{i+1} doivent être mises à jour et la nouvelle politique de t_i doit donc être connue. Afin d'estimer correctement le coût occasionné par E sur G, la nouvelle politique de F doit par conséquent être connue. Ceci n'est malheureusement pas possible puisque les politiques de E et F sont révisées simultanément. L'agent 1 utilise donc les probabilités sur les taux de ressources précédemment calculées à partir des politiques initiales, obtenant alors une mesure du coût occasionné pouvant être incorrecte. Supposons que la politique initiale de l'agent 2 soit de commencer l'exécution de F au temps 12 et que celle de l'agent 1 consiste à exécuter E à 13. Admettons qu'une telle politique conduise fort probablement l'agent 2 à échouer partiellement l'exécution de F et à manquer de ressources pour l'exécution de G. Le coût occasionné sur G sera alors faible quel que soit le retard puisque la probabilité que G puisse s'exécuter est faible. Ainsi, lors de la révision des politiques d'exécution de E et F, l'agent 1 considère le coût occasionné sur G comme négligeable et peut choisir de retarder l'exécution de E (par exemple à 15). En effet, puisque l'agent 2 ne possède pas assez de ressources pour exécuter G, l'agent 1 peut retarder G autant qu'il le souhaite même si cela implique que les contraintes temporelles de G ne puissent pas être respectées. L'agent 2 va de son côté retarder l'exécution de F afin de limiter les échecs partiels et leur consommation de ressources. Ainsi, les ressources disponibles avant l'exécution de G vont augmenter et la tâche G pourra être exécutée. Le coût occasionné utilisé par l'agent 1 afin de déterminer la politique d'exécution de E va alors devenir inexact. L'agent 1 a en effet choisi de retarder E car le coût occasionné sur G était négligeable. Du fait de la modification de la politique d'exécution de F, ce coût est en réalité plus important. La décision prise par l'agent 1 peut alors s'avérer inappropriée et conduire à une diminution des performances des agents. En raison des variations dans les disponibilités des ressources, l'utilité espérée du système est donc susceptible de diminuer et la convergence ne peut être garantie.

Mesures du coût occasionné et convergence

L'utilisation d'une méthode d'estimation du coût occasionné qui ne tienne pas compte des ressources peut, à partir de ces constatations, s'avérer pertinente. La mesure qui est alors réalisée demeure dans ce cas correcte quelles que soient les variations sur les disponibilités des ressources. En revanche, la valeur du coût occasionné ainsi calculée risque de s'avérer moins précise que celle obtenue en considérant les ressources.

Une telle méthode de calcul a été décrite en section 7.4. Le coût occasionné est alors déterminé en considérant les ressources des autres agents comme maximales ce qui correspond à une vision optimiste des possibilités des autres agents. Nous avons précédemment discuté des répercussions d'une telle mesure sur l'estimation du coût occasionné (sur-estimation et sous-estimation). Nous avons alors montré que cette méthode pouvait conduire, du fait des erreurs d'estimation, à des solutions de moins bonne qualité, comparée à la méthode tenant compte des ressources. Cette mesure permet néanmoins de garantir la convergence de l'algorithme sous certaines hypothèses. En raison de leur indépendance vis à vis des ressources, les valeurs de coût occasionné demeurent en effet exactes même si les disponibilités des ressources sont modifiées. Ainsi, lorsque les tâches d'un même niveau sont considérées simultanément, nous pouvons garantir que le coût occasionné est correctement estimé.

Méthode de calcul du coût occasionné	coût occasionné espéré dépendant des ressources	coût occasionné espéré indépendant des ressources	coût occasionné indépendant des ressources
Précision	Oui	Non	Non
Convergence	Non	Oui	Oui

TAB. 9.1 – Influence des méthodes de calcul du coût occasionné sur la convergence

Le tableau 9.1 met en relation les différentes méthodes de calcul du coût occasionné que nous avons précédemment proposées, en indiquant pour chacune d'elles si elle fournit une mesure précise du coût occasionné et si elle permet de garantir la convergence de l'algorithme itératif lorsque la politique EST est utilisée comme politique initiale. L'utilisation d'une méthode de calcul précise du coût occasionné nécessite de prendre en compte les ressources et ne peut donc pas assurer la convergence. Néanmoins, dans le cas où l'algorithme converge, la solution est de meilleure qualité que celles obtenues par les autres méthodes d'estimation du coût occasionné. En effet, toute politique est calculée en utilisant une estimation précise de l'influence des actions sur les autres agents.

Cette comparaison nous amène à suggérer l'utilisation d'une approche mixte conjugant les différentes méthodes de calcul du coût occasionné qui ont été proposées. Ainsi, dans le cas où l'utilisation d'une mesure précise (tenant compte des ressources) risque de mener à une divergence de l'algorithme, nous proposons d'avoir recours à une autre méthode de calcul du coût occasionné. Sinon, nous utilisons la méthode de calcul la plus précise. Afin de déterminer le coût occasionné par l'exécution d'une tâche t_i sur un agent $\mathcal{A}g_j$, nous utilisons ainsi la méthode basée sur les ressources à condition qu'aucune tâche du même niveau que t_i ne soit exécutée par $\mathcal{A}g_j$. Sinon, nous utilisons la méthode basée sur une mesure du coût occasionné espéré indépendante des

ressources. Nous pouvons ainsi augmenter la précision de la mesure et améliorer la qualité des politiques tout en garantissant la convergence.

La révision simultanée des politiques des agents est à l'origine des cas de divergence de notre algorithme. Néanmoins, elle est également un des facteurs de son efficacité. Ainsi, des algorithmes comme JESP [Nair *et al.*, 2003] garantissent la convergence du processus itératif mis en place, mais se trouvent en contrepartie limités par la taille des problèmes pouvant être traités.

La version centralisée de notre algorithme considère les tâches d'un niveau L_i l'une après l'autre. Comme nous ne tirons pas parti de cette séquentialité, le procédé est équivalent à une révision simultanée des politiques d'exécution des tâches. Nous pourrions envisager profiter de cette séquentialité et tenir compte, lors de la révision d'une politique t_i appartenant à L_i, des modifications déjà réalisées sur les politiques d'exécution des tâches du niveau L_i. Ceci nécessiterait cependant de réviser le modèle de transition chaque fois qu'une politique est modifiée et diminuerait l'efficacité de notre algorithme. Ce dernier n'effectue en effet qu'une seule révision des fonctions de transition à chaque itération. Une telle mise à jour, à chaque modification d'une politique, risquerait d'augmenter fortement la complexité temporelle de la résolution.

Remarquons que dans les cas où l'algorithme diverge, nous sommes toutefois en mesure de fournir une solution au problème de planification qui nous est posé. En effet, chaque itération résulte en un ensemble de politiques correspondant à une politique jointe et constituant une solution au problème.

Il est par ailleurs envisageable de détecter les points de divergence en comparant l'utilité des politiques jointes obtenues à la fin de chaque itération.

9.3 A propos de l'optimalité

9.3.1 Cas particuliers d'optimalité

Nous avons mis en évidence, au chapitre précédent, un certain nombre de propriétés garantissant l'optimalité de la solution retournée par notre algorithme de révision des politiques. Ces propriétés garantissent, de la même façon, l'optimalité de la solution établie par l'algorithme itératif. Nous pouvons de plus affirmer qu'une seule itération est nécessaire à l'algorithme pour trouver la solution optimale. Deux itérations auront cependant lieu, la seconde étant destinée à vérifier que la solution ne peut plus être améliorée. En effet, à la première itération, nous appliquons l'algorithme de révision des politiques qui retourne la solution optimale. Une fois cette solution établie, l'algorithme est re-exécuté. Il n'est alors plus possible d'améliorer la politique d'exécution d'une quelconque tâche, l'ensemble des politiques n'est donc pas modifié et le point de convergence est atteint.

Par ailleurs, le théorème 8 peut être généralisé à un ensemble de politiques initiales quelconques :

Théorème 9 *Si le graphe de la mission ne comprend que deux agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$ et que toutes les contraintes de précédence sont orientées de $\mathcal{A}g_i$ vers $\mathcal{A}g_j$ alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.*

Preuve : Si la politique initiale de l'agent $\mathcal{A}g_i$ est la politique EST, nous avons précédemment démontré que la politique optimale jointe était calculée en une itération. Lorsque l'ensemble des politiques initialement fixées est quelconque, la politique de l'agent $\mathcal{A}g_i$ calculée à la première itération est la politique EST. La seconde itération permet alors de déterminer la politique optimale jointe (pour plus de détails, se reporter à la preuve du théorème 8). □

9.3.2 Situation d'équilibre

Dans le cas général où aucune des propriétés recensées en section 8.3.2 n'est vérifiée, nous ne pouvons garantir l'optimalité de la solution. Développer un algorithme qui apporterait de telles garanties conduirait à une complexité trop importante de la méthode de résolution et ne permettrait pas de résoudre les problèmes envisagés.

L'algorithme itératif proposé s'arrête lorsque la politique d'exécution de chaque tâche ne peut plus être améliorée. Ainsi, étant données les politiques des autres agents, aucun agent n'est en mesure de modifier sa politique de sorte que les performances du système augmentent. Nous sommes donc dans une situation d'équilibre.

Conclusion

Nous avons, dans ce chapitre, présenté une version itérative de l'algorithme de révision décrit au chapitre 8. Cet algorithme itératif renouvelle le processus de révision des politiques en considérant l'ensemble des politiques finales de l'itération $N-1$ comme étant l'ensemble initialement fixé à l'itération N. A chaque itération, l'utilisation de l'algorithme de révision des politiques permet de limiter la fréquence des mises à jour du modèle. En effet, seules les fonctions de transition doivent être re-calculées au début de chaque itération, aucune autre modification du modèle n'étant nécessaire.

Nous avons décrit deux versions de l'algorithme itératif, chacune basée sur l'une ou l'autre des versions (centralisée ou décentralisée) de l'algorithme de co-évolution. Nous avons ensuite étudié la complexité de ces deux versions et avons plus particulièrement traité de l'influence du nombre d'itérations sur la complexité temporelle de l'algorithme. Ceci nous a amené à discuter de sa convergence. Nous avons alors montré que les garanties de convergence de l'algorithme itératif dépendaient de la méthode d'estimation du coût occasionné à laquelle nous avons recours.

Enfin, nous avons traité de l'optimalité des solutions calculées. Il a ainsi été mis en évidence que les propriétés précédemment recensées et menant à une politique optimale dans le cas de l'algorithme de révision, garantissaient également l'obtention d'une politique optimale lors de l'exécution de l'algorithme itératif. Dans le cas général, nous avons prouvé que la politique jointe calculée correspond à une situation d'équilibre.

Conclusion de la partie III

Dans cette troisième partie, nous nous sommes intéressée à la résolution du problème de planification multi-agent que nous avions formalisé par un OC-DEC-MDP.

Nous avons tout d'abord finalisé cette formalisation en proposant une méthode de décomposition de la fonction de transition du système. Nous avons pour cela fixé les politiques des agents et avons calculé les probabilités temporelles sur l'exécution de chaque tâche. A partir de ces probabilités, nous avons déduit la fonction de transition propre à chaque agent, en prenant soin de modéliser les contraintes sur l'exécution des tâches. Une fois les fonctions de transition individuelles déterminées, nous avons été en mesure de définir de manière complète l'ensemble des MDPs constituant l'OC-DEC-MDP considéré.

Nous nous sommes alors intéressée, dans le chapitre suivant, à la résolution de cet ensemble de MDPs. Afin d'optimiser les performances du système, nous avons mis en évidence la nécessité de mettre en place un mécanisme de coordination entre les agents. Pour ce faire, nous nous sommes inspirée d'une notion provenant des sciences économiques : le coût occasionné. Ce dernier permet à chaque agent de mesurer l'influence de ses actions sur les autres agents, et plus particulièrement la perte en utilité induite sur les autres agents. La politique d'un agent $\mathcal{A}g_i$ à partir d'un état s_i, résulte ainsi d'un compromis entre l'utilité espérée de $\mathcal{A}g_i$ et le coût occasionné. Il a été démontré que la politique alors obtenue conduit à un comportement coopératif des agents et maximise l'utilité espérée du système.

Nous avons par la suite cherché à définir un algorithme utilisant ce procédé de planification et calculant la politique de chaque agent $\mathcal{A}g_i$ en tout état s_i. Afin de garantir l'efficacité de la planification, nous nous sommes orientée vers la recherche d'une approximation de la solution optimale. Nous avons également cherché à limiter la fréquence des révisions du modèle et les

communications entre les agents[46].

L'algorithme que nous avons développé procède par améliorations successives d'un ensemble de politiques initiales, en ordonnançant la révision des politiques d'exécution de chaque tâche. De complexité polynomiale, cet algorithme permet d'envisager la résolution de problèmes de grande taille. Il gère par ailleurs la modélisation des contraintes temporelles et de ressources. Enfin, il garantit, sous certaines conditions, l'optimalité de la solution calculée.

Nous avons enfin souhaité améliorer la qualité de la solution calculée par le précédent algorithme, lorsque l'optimalité n'est pas garantie. Nous avons ainsi décrit une version itérative de l'algorithme qui renouvelle le processus de révision des politiques et conduit, en cas de convergence, à un équilibre. A chaque itération, les politiques des agents sont révisées en parallèle ce qui permet d'augmenter l'efficacité de la planification. Ceci peut, en contrepartie, mener à des cas de divergence de l'algorithme. Après avoir identifié ces cas, des solutions ont été proposées afin de rester dans un cadre où la convergence est assurée.

En raison du lien étroit entre qualité de la solution et efficacité de l'algorithme, nous allons, dans la suite de ce document, analyser ces deux aspects de notre approche. Nous nous intéresserons à son efficacité tout en étudiant la taille des problèmes que nous pouvons raisonnablement traiter. Nous nous intéresserons ensuite à la qualité des solutions calculées par les deux algorithmes qui ont été présentés. Nous nous pencherons alors plus particulièrement sur l'influence de la mesure du coût occasionné sur la qualité de la solution.

[46] dans le cas d'une exécution décentralisée de l'algorithme

Quatrième partie

Validation de l'approche

Introduction de la partie IV

Notre travail a eu pour objectif la mise en place d'un processus de planification multi-agent pouvant être utilisé afin de résoudre des problèmes décisionnels tels que ceux rencontrés en robotique exploratoire. Pour ce faire, nous avons mis en évidence la nécessité de développer une approche permettant de gérer les incertitudes et de tenir compte de différents types de contraintes sur l'exécution des tâches. L'absence de communication et la limitation des calculs durant l'exécution des tâches ont également été prises en compte.

La deuxième partie de ce document a concerné la modélisation de ces problèmes sous forme d'OC-DEC-MDP. Nous avons ainsi pu prendre en compte les incertitudes et formaliser les différents types de contraintes qui avaient été recensés. Nous nous sommes ensuite orientée vers le calcul de politiques individuelles permettant aux agents de décider, facilement et sans communiquer, comment agir. Les problèmes envisagés pouvant être de taille importante, il a été nécessaire de mettre en place un algorithme de résolution efficace. Ce sujet a été traité dans la partie précédente de ce document au cours de laquelle deux algorithmes permettant de calculer une approximation de la politique optimale ont été présentés.

Nous allons, dans cette partie, chercher à vérifier que les solutions qui ont été précédemment proposées répondent bien aux exigences initialement recensées au chapitre 4. Nous allons en particulier tester l'efficacité de notre approche et étudier la qualité des solutions obtenues. Nous nous intéresserons ensuite à l'applicabilité de notre travail. A partir d'un scénario de la robotique exploratoire, nous vérifierons alors que les solutions proposées permettent la prise en compte des incertitudes et des contraintes, et qu'elles peuvent être utilisées par des robots en situation réelle d'exécution d'une mission.

Chapitre 10

Expérimentations

L'étude du contexte applicatif de notre travail a mis en évidence la nécessité de gérer des missions composées d'une centaine de tâches et d'une dizaine d'agents. Étant donnée la taille de ces missions, nous avons choisi de mettre en place une approche décomposant le problème et calculant une approximation de la solution optimale. Il a ainsi été possible de développer des algorithmes de complexité moins importante que ceux proposés jusqu'à maintenant. L'étude de la complexité ainsi que l'examen des cas d'optimalité des deux algorithmes que nous avons présentés promettent des résultats à la hauteur de nos espérances.

Nous allons, dans ce chapitre, développer différentes expérimentations permettant d'étudier et de tester l'efficacité de notre approche ainsi que la qualité des solutions obtenues. Des expérimentations seront menées sur chacun des algorithmes précédemment présentés. Nous nous intéresserons plus particulièrement à la taille des problèmes pouvant être envisagés, au temps de résolution requis et à la qualité des politiques calculées. Après avoir réalisé cette étude pour l'algorithme de révision des politiques et pour l'algorithme itératif, nous nous pencherons sur les apports de la version itérative et sur l'influence de la mesure du coût occasionné sur les résultats obtenus. Ces expérimentations nous permettront ainsi de conclure sur l'adéquation entre notre approche et les attentes initialement formulées.

10.1 Évolution du nombre d'états

Nous avons précédemment démontré que la complexité de l'algorithme de révision des politiques, que ce soit dans sa version centralisée ou décentralisée, est étroitement liée à la taille de l'espace d'états des agents. Nous allons de ce fait commencer par définir les différents paramètres d'une mission intervenant dans la définition des états du modèle et nous étudierons leur influence sur le nombre d'états. Nous pourrons ainsi mettre en évidence les différentes composantes d'une mission influençant la complexité de l'algorithme.

Chaque état étant défini à partir d'une tâche, d'un intervalle et d'un taux de ressources, les

trois paramètres que sont le nombre de tâches, le nombre d'intervalles par tâche et le nombre de taux de ressources possibles par tâche, influencent directement la taille de l'espace d'états de chaque agent, et donc la complexité du problème. Afin d'envisager plus précisément les capacités de notre approche, nous allons donc commencer par étudier l'impact des données du problème sur ces paramètres et sur le nombre d'états.

10.1.1 Nombre de tâches

Le nombre de tâches que chaque agent doit réaliser influence naturellement le nombre d'états. Plus un agent doit exécuter de tâches, plus son espace d'états est important.

Le nombre de tâches par agent est étroitement lié au nombre d'agents impliqués dans la mission. Ainsi, pour un même ensemble de tâches, le nombre d'états varie en fonction du nombre d'agents considérés. Les figures 10.1 et 10.2 décrivent l'évolution de la taille[47] de l'espace d'états d'un agent, en fonction du nombre d'agents impliqués dans l'exécution d'un même ensemble de tâches. Des missions de 20, 50, 100, 150 et 200 tâches sont considérées. Deux phases peuvent alors être identifiées en ce qui concerne l'évolution du nombre d'états de chaque agent. La première phase, absente pour les missions de 150 et 200 tâches, consiste en une brusque augmentation du nombre d'états. Elle est suivie par une phase de diminution du nombre d'états au cours de laquelle plus le nombre d'agents augmente, plus la taille de l'espace d'états décroît.

La brusque hausse du nombre d'états constituant la première phase s'explique par l'augmentation des dépendances entre les agents. En effet, augmenter le nombre d'agents conduit à re-répartir les tâches entre les agents. Les agents déjà présents sont ainsi délestés de certaines tâches qui sont attribuées aux nouveaux agents. Des tâches exécutées par un même agent se trouvent alors réalisées par des agents différents d'où une augmentation des dépendances entre les agents. Considérons deux tâches t_i et t_{i+1} exécutées par un agent $\mathcal{A}g_i$ et telles que t_i soit un prédécesseur de t_{i+1}. $\mathcal{A}g_i$ étant chargé de ces deux tâches, nous pouvons garantir que l'exécution de t_i sera terminée lorsque $\mathcal{A}g_i$ tentera d'exécuter t_{i+1}. Si nous ajoutons un nouvel agent $\mathcal{A}g_j$ à la mission, il est possible de délester $\mathcal{A}g_i$ d'une de ces deux tâches en attribuant, par exemple, la tâche t_i à $\mathcal{A}g_j$. La contrainte de précédence entre t_i et t_{i+1} traduit alors un lien de dépendance entre les agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$. Bien que le nombre de tâches de $\mathcal{A}g_i$ diminue, son nombre d'états d'échec partiel augmente puisque l'exécution de t_{i+1} dépend à présent de l'exécution de t_i par un autre agent. L'augmentation du nombre d'agents conduit donc, dans un premier temps à une hausse des dépendances entre agents qui se traduit par une augmentation du nombre d'états d'échecs partiels. Puis, le nombre de dépendances entre les agents stagne : si nous ajoutons un troisième agent $\mathcal{A}g_k$ à notre exemple et que nous lui attribuons la tâche t_{i+1}, le nombre de dépendances entre agents reste le même. L'ajout d'agents se traduit alors uniquement par une baisse

[47] Moyenne réalisée sur tous les agents.

du nombre de tâches par agent, d'où une diminution du nombre d'états de succès [tâche, intervalle, ressource] et du nombre d'états d'échec partiel [tâche, intervalle, date de fin, ressource]. Ce phénomène caractérise ainsi la seconde phase précédemment identifiée.

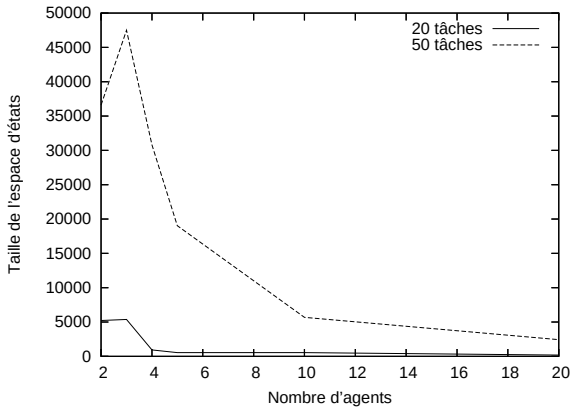


FIG. 10.1 – Évolution de l'espace d'états en fonction du nombre d'agents

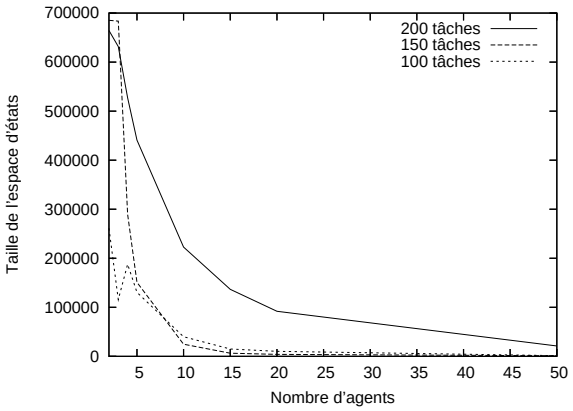


FIG. 10.2 – Évolution de l'espace d'états en fonction du nombre d'agents

La figure 10.3 détaille l'évolution de chacune des catégories d'états (états de succès, états d'échec partiel et états d'échec total) à partir du graphe de 50 tâches utilisé lors des expérimentations décrites par la figure 10.2. Nous constatons que l'augmentation du nombre d'agents se traduit par une baisse du nombre d'états de succès. En effet, le nombre de tâches par agent diminuant, le nombre de triplets [tâche, intervalle, ressource] décroît. Nous constatons en revanche, une hausse du nombre d'échecs partiels s'expliquant par l'augmentation des dépendances entre les agents. Cette hausse est répercutée sur le nombre total d'états ce qui explique les « pics » constatés sur les figures 10.1 et 10.2. La figure 10.4 détaille, quant à elle, l'évolution des différents types d'états à partir du graphe composé de 150 tâches et utilisé lors des expérimentations décrites par la figure 10.1. Dans ce cas, la hausse du nombre d'échecs partiels est masquée par la diminution du nombre d'états de succès. Le nombre de tâches de la mission étant important, l'ajout d'un agent entraîne en effet une baisse importante du nombre de tâches par agent. La baisse du nombre d'états de succès qui s'en suit étant alors plus importante que la hausse du nombre d'échecs partiels, nous assistons à une diminution du nombre d'états.

Les figures 10.5 et 10.6 décrivent enfin l'évolution du nombre total d'états (somme du nombre d'états de chaque agent) en fonction du nombre d'agents. Nous retrouvons le même profil d'évolution que pour les expérimentations qui précèdent. Ainsi, nous observons que le nombre d'états par agent et le nombre total d'états évoluent de façon similaire lorsque le nombre d'agents impliqués

dans une mission augmente.

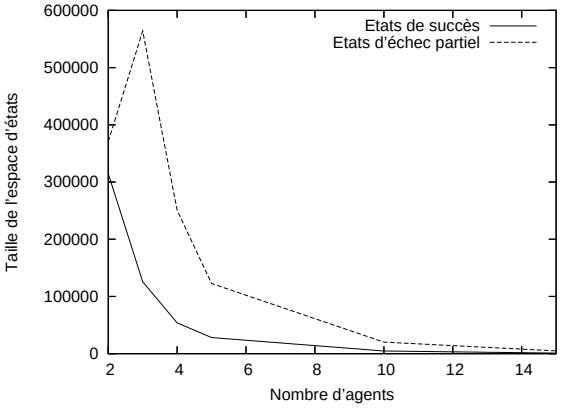


FIG. 10.3 – Évolution des différents types d'états en fonction du nombre d'agents

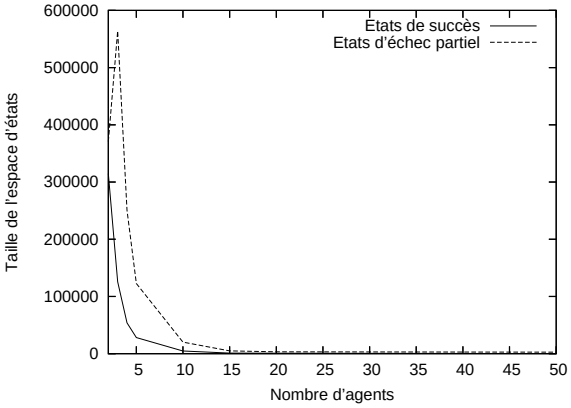


FIG. 10.4 – Évolution de l'espace d'états en fonction du nombre d'agents

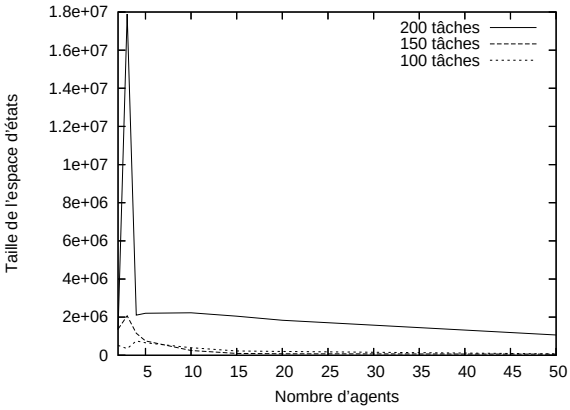


FIG. 10.5 – Évolution du nombre total d'états en fonction du nombre d'agents

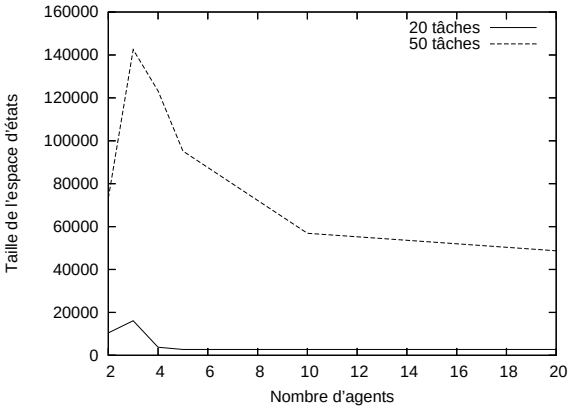


FIG. 10.6 – Évolution du nombre total d'états en fonction du nombre d'agents

10.1.2 Nombre d'intervalles par tâche

Les intervalles d'exécution constituent la seconde composante des états de succès. Le nombre d'intervalles par tâche influence ainsi directement le nombre d'états de succès associés à chaque tâche, et par conséquent la taille des espaces d'états. Les intervalles d'exécution définissent également les dates de fin d'une tâche. Ces dernières étant prises en compte dans la description des états d'échec partiel, le nombre d'intervalles conditionne donc aussi le nombre d'états d'échec partiel de chaque agent.

Nous avons présenté au chapitre 5 une méthode permettant de déterminer l'ensemble des intervalles d'exécution des tâches constituant la mission. A partir des calculs qui ont alors été décrits, nous pouvons affirmer que le nombre d'intervalles associés à une tâche t_i dépend :

- des contraintes temporelles de t_i (date de début au plus tôt et date de fin au plus tard),
- des contraintes de précédence liées à t_i,
- des durées d'exécution de t_i.

Le nombre d'intervalles possibles au pire cas pour t_i est alors donné par :

$$\#I = (LET - \delta_c^{i,min} - EST_i) \times \#_{duree} \tag{10.1}$$

où $\#_{duree}$ désigne le nombre de durées possibles pour t_i et $\delta_c^{i,min}$ correspond à la durée minimale d'exécution de t_i.

Contraintes temporelles

L'équation 10.1 met en évidence l'influence des contraintes temporelles d'une tâche t_i sur le nombre d'intervalles de t_i. Ainsi, lorsque l'amplitude entre EST et LET est faible, les possibilités d'exécution de t_i sont restreintes et le nombre d'intervalles d'exécutions associés à la tâche est faible. Lorsque la taille des fenêtres temporelles $[EST_i, LET_i]$ augmente, nous constatons une augmentation de la taille de l'espace d'états. En effet, les contraintes temporelles sont relâchées, un plus grand nombre d'exécutions est ainsi possible pour la tâche, il s'en suit donc une augmentation du nombre d'états de succès et d'échec partiel. La figure 10.9 illustre cette évolution sur une mission d'une centaine de tâches. L'échelle des contraintes temporelles traduit l'amplitude des fenêtres $[EST, LET]$. Pour chaque tâche de la mission, une fenêtre temporelle est initialement attribuée. Celle-ci correspond à une échelle de 1. Pour une échelle de 2, la même mission est considérée mais la taille de chaque fenêtre temporelle est doublée. Ainsi, une échelle de 1.5 désigne une mission pour laquelle l'amplitude des fenêtres temporelles $[EST, LET]$ a été multipliée par 1.5. La figure 10.9 montre que l'augmentation de la taille des fenêtres temporelles conduit tout d'abord à une hausse du nombre d'états. Puis, la taille de l'espace d'états reste constante malgré le relâchement des contraintes temporelles. En effet, au fur et à mesure de la relaxation des contraintes, un plus grand nombre d'intervalles d'exécution est envisageable.

Lorsque le pallier est atteint et que le nombre d'états ne change plus, nous sommes dans une situation où les fenêtres temporelles sont tellement larges qu'elles ne contraignent plus l'exécution des tâches. Tous les intervalles d'exécution envisageables sont alors considérés. Continuer à relâcher les contraintes n'entraîne plus la prise en compte d'intervalles supplémentaires et le nombre d'états reste constant.

Contraintes de précédence

Les contraintes de précédence constituent le second facteur influençant le nombre d'intervalles d'exécution d'une tâche t_i. Bien que l'influence de ces contraintes ne soit pas directement visible dans l'équation 10.1, elle a déjà été évoquée précédemment et mise en évidence lors du calcul des dates de début décrit au chapitre 5. Les contraintes de précédence influencent en effet le nombre de dates de début d'une tâche t_i puisque cette dernière ne peut commencer que si ses prédécesseurs ont terminé leur exécution. Le nombre d'intervalles d'exécution étant directement lié au nombre de dates de début possibles, la taille de l'espace d'états dépend des contraintes de précédence.

L'ajout de contraintes de précédence peut conduire à une augmentation ou à une diminution de la taille de l'espace d'états. En effet, l'ajout d'une contrainte de précédence au niveau d'une tâche t_i augmente d'une unité le nombre de prédécesseurs de t_i. L'ensemble des intervalles de t_i est alors, suivant les configurations, restreint ou accru. La figure 10.7 décrit un cas où l'ajout d'un lien de précédence conduit à une augmentation du nombre de dates de début et d'intervalles de C, impliquant une augmentation de l'espace d'états de l'agent 1. En revanche, la figure 10.8 illustre une situation où un tel ajout suscite une baisse du nombre d'intervalles de C et par conséquent une diminution de l'espace d'états de l'agent 1. Cette diminution est due à la contrainte temporelle sur la date de fin de C. Toute date de début supérieure à 5 n'est en effet pas envisageable puisqu'elle conduit avec certitude à la violation des contraintes temporelles.

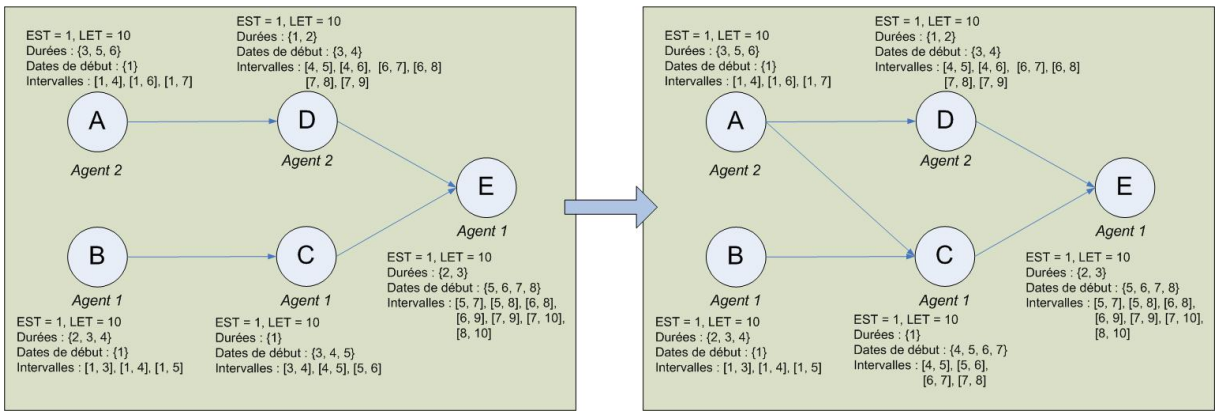


FIG. 10.7 – Ajout d'une contrainte temporelle conduisant à l'ajout de dates de début

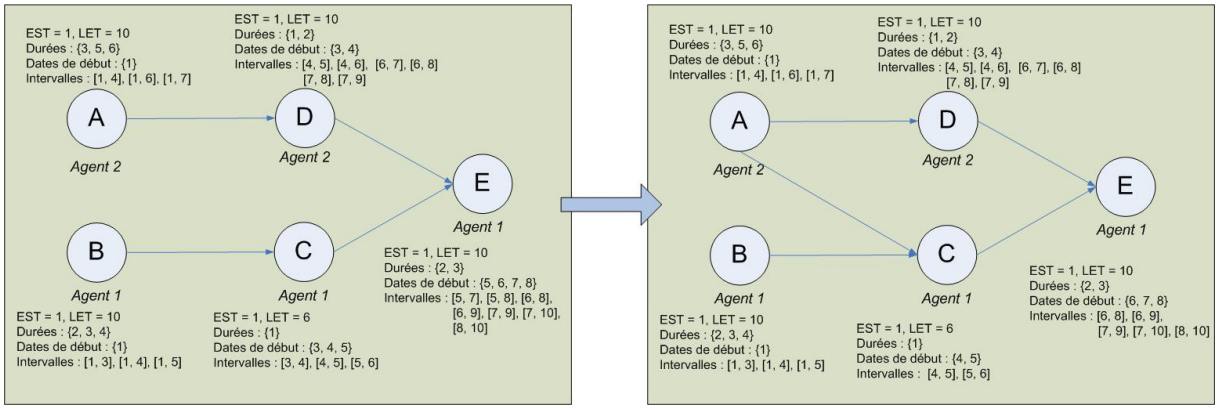


FIG. 10.8 – Ajout d'une contrainte temporelle conduisant à la suppression de dates de début

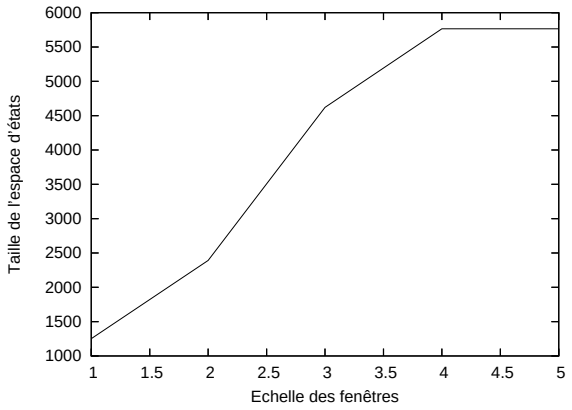


FIG. 10.9 – Évolution de l'espace d'états en fonction de la taille des fenêtres temporelles

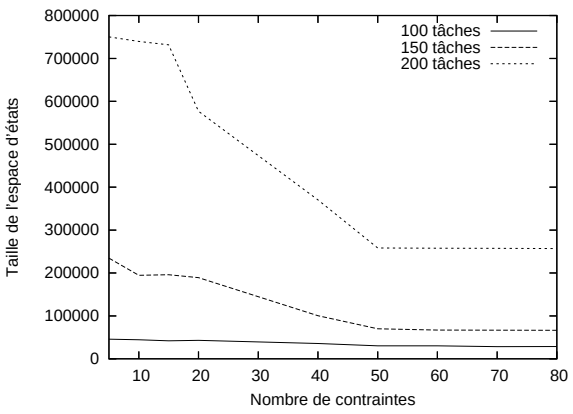


FIG. 10.10 – Evolution en fonction du nombre de contraintes

Une étude plus approfondie de la progression du nombre d'intervalles en fonction du nombre de contraintes de précédence a permis d'identifier trois phases d'évolution distinctes, chacune caractérisée par un profil d'évolution particulier. La première consiste en une phase d'oscillation au cours de laquelle l'ajout de contraintes conduit à l'augmentation où la diminution du nombre d'intervalles donc du nombre d'états. La seconde phase se caractérise par une diminution du nombre d'intervalles. La troisième phase consiste enfin en une stagnation du nombre d'intervalles. Ces trois phases se succèdent toujours dans cet ordre avec une plus ou moins grande amplitude. Durant la première phase, l'ajout d'une contrainte de précédence à une tâche t_i peut mener à l'ajout de dates de début non encore considérées (augmentation du nombre d'intervalles) ou bien retarder la date de début au plus tôt de la tâche t_i (diminution du nombre d'intervalles). Au fur

et à mesure de l'ajout de telles contraintes, nous nous rapprochons d'un point de « saturation » où toutes les dates de début possibles de chaque tâche ont été ajoutées. Ajouter une nouvelle contrainte ne peut alors que retarder la dernière date de début (UB) et conduit à une diminution du nombre de dates de début. La troisième phase se traduit enfin par une stagnation du nombre de dates de début, donc d'états. En effet, toutes les dates de début possibles ont été envisagées et leur ensemble a été restreint au maximum, l'ajout de contraintes de précédence ne modifie donc pas cet ensemble.

La figure 10.10 illustre cette évolution sur différentes tailles de mission. Considérons la mission composée de 150 tâches : l'ajout de contraintes de précédence conduit tout d'abord à des oscillations (jusqu'à 20 contraintes). Puis, ces ajouts se traduisent par une baisse du nombre d'états. Enfin, nous assistons à un stagnation de la taille de l'espace d'états. Cette évolution est remarquable aussi bien au niveau de l'espace d'états complet de chaque agent, qu'au niveau du nombre d'états de succès ou d'échec partiel. Le nombre de dates de début influence en effet de façon similaire le nombre d'états de succès et le nombre d'états d'échec partiel.

Durées d'exécution

Le nombre d'intervalles d'exécution d'une tâche t_i est enfin conditionné par le nombre de durées d'exécution de t_i. Augmenter le nombre de durées d'exécution revient à affiner la discrétisation et nous rapproche d'une modélisation continue du temps d'exécution. Plus il existe de durées différentes pour l'exécution de t_i, plus le nombre d'intervalles de cette tâche est important. Le nombre d'intervalles est toutefois limité par les contraintes temporelles de t_i qui imposent une borne supérieure sur les dates de fin possibles de la tâche. Ainsi, les intervalles d'exécution $I = [st, et]$ de t_i tels que $st + \delta_c^i = et$ et $et > LET_i$ ne peuvent pas être considérés comme des intervalles d'exécution valides et ne sont donc pas envisagés lors de la construction de l'espace d'états.

De nombreux paramètres entrent ainsi en jeu dans l'évolution du nombre d'intervalles d'exécution d'une tâche t_i. Relaxer les contraintes temporelles ou augmenter le nombre de durées d'exécution mène à une hausse du nombre d'intervalles. En revanche, l'ajout de contraintes de précédence peut augmenter ou diminuer le nombre d'intervalles. De manière plus générale, une hausse du nombre d'intervalles conduit de manière certaine à une augmentation de la taille de l'espace d'états. Réciproquement, une baisse du nombre d'intervalles d'exécution entraîne une diminution du nombre d'états.

10.1.3 Nombre de taux de ressources par tâche

Avant toute exécution d'une mission, chaque agent dispose d'une quantité R_{ini} de ressources qu'il consomme au fil de la réalisation de ses tâches et de ses échecs partiels. Cette quantité de ressources initiale, ainsi que le nombre de consommations de ressources possibles pour chaque tâche, constituent les derniers paramètres de la mission ayant une influence sur le nombre d'états. Chaque taux de ressources associé à un état s_i dépend, en effet, des ressources initialement disponibles par l'agent ainsi que des ressources qui ont été consommées.

La quantité de ressources disponible initialement pour un agent $\mathcal{A}g_i$ est à mettre en relation avec les tâches devant être exécutées par $\mathcal{A}g_i$. Ainsi, une même quantité initiale de ressources peut s'avérer plus ou moins suffisante suivant le nombre de tâches que l'agent doit effectuer et les consommations de ces tâches. Lorsque le nombre d'agents impliqués dans une mission augmente, le nombre de tâches par agent diminue. Si la quantité de ressources initiale reste identique pour chaque agent, les ressources deviennent alors de plus en plus importantes par rapport au travail à réaliser.

Les figures 10.11 et 10.12 reprennent les missions de 200 tâches et 20 tâches décrites par les figures 10.1 et 10.2. Nous avons cependant fait varier les ressources initialement disponibles par chaque agent et avons pu remarquer une atténuation, jusqu'à sa complète disparition, du « pic » précédemment mis en évidence par l'étude de l'influence du nombre d'agents sur la taille des espaces d'états. Nous avons également pu constater que la présence de ce pic coïncide avec des ressources initialement insuffisantes afin d'exécuter le travail demandé. Il semble donc que la hausse du nombre d'états ne soit pas seulement due à une augmentation du nombre des dépendances entre les agents. Il apparaît en effet qu'elle est également liée à la quantité de ressources initialement disponible. Comme le montre la figure 10.12, une quantité initialement faible de ressources par rapport aux tâches à exécuter s'accompagne, lorsque le nombre d'agents augmente, d'un pic dans le nombre d'états. En effet, lorsque peu d'agents sont engagés dans la réalisation de la mission, leur probabilité de manquer de ressources est forte. Il existe alors peu de taux de ressources positifs pouvant être disponibles après l'exécution de chaque tâche. L'augmentation du nombre d'agents réduit la charge de chacun, les ressources deviennent alors plus importantes par rapport au travail demandé et un plus grand nombre de taux de ressources peut être envisagé pour chaque tâche. Cette hausse du nombre de taux de ressources possibles pour chaque tâche conduit alors à une augmentation du nombre d'états. Lorsque les ressources sont suffisantes pour exécuter toutes les tâches et échouer partiellement, les agents ne sont plus amenés à manquer de ressources. Tous les combinaisons de consommations de ressources conduisent à des taux de ressources positifs et sont donc envisagées. Continuer à augmenter les ressources en augmentant le nombre d'agents, n'entraîne alors plus la prise en compte de nouvelles combinaisons de consommations de ressources et le nombre de taux de ressources possibles pour chaque tâche

reste constant. L'ajout d'agents se traduit alors uniquement par une baisse du nombre de tâches par agent et donc par une diminution du nombre d'états de succès et d'échecs partiels de chaque agent.

Nous avons précédemment montré que la hausse du nombre d'agents suscite une augmentation des dépendances entre les agents se traduisant par une augmentation du nombre d'échecs partiels. Lorsque les ressources du problème sont importantes, cette hausse est masquée par la baisse des taux de ressources possibles pour chaque agent. Le nombre d'échecs partiels diminue donc au fur et à mesure que des agents sont ajoutés à la mission.

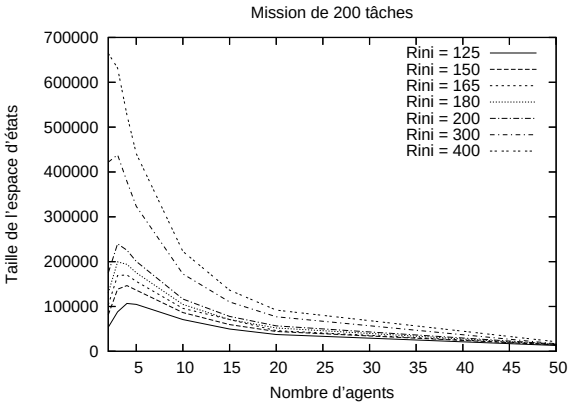


FIG. 10.11 – Évolution de l'espace d'états en fonction de R_{ini} et du nombre d'agents

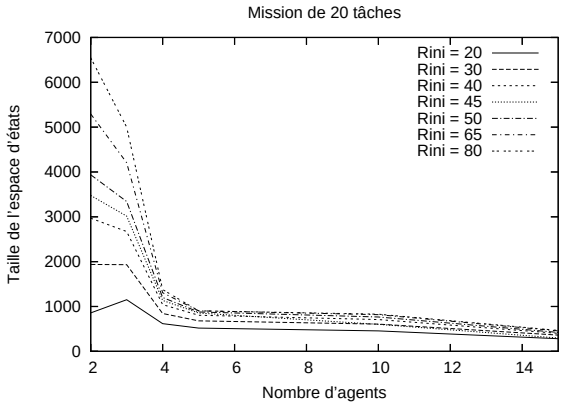


FIG. 10.12 – Évolution de l'espace d'états en fonction de R_{ini} et du nombre d'agents

Comme nous venons de le montrer, un grand nombre de paramètres entrent en jeu dans la définition de l'espace d'états de chaque agent. Il est donc difficile d'estimer *a priori* le nombre d'états qu'il sera nécessaire d'évaluer afin de déterminer les politiques des agents. Les expérimentations qui viennent d'être présentées permettent néanmoins d'affirmer que les contraintes temporelles et de précédence contribuent à limiter la taille des espaces d'états en restreignant les intervalles d'exécution possibles pour chaque tâche. De la même façon, la limitation des ressources permet de restreindre le nombre d'états associés à chaque tâche.

Malgré les nombreux paramètres influençant le nombre d'états d'un agent, ces expérimentations ont toutefois permis d'estimer l'ordre de grandeur des espaces d'états qui doivent être traités par nos algorithmes. Ainsi, une mission composée de 200 tâches et 3 agents nous a conduit à obtenir des MDPs de l'ordre de 250 000 états. Lorsque les ressources initiales augmentent, cette taille peut être portée à 700 000 états. La présence d'un grand nombre de contraintes de précédence

et d'agents permet d'augmenter le nombre de tâches à exécuter tout en conservant un nombre d'états raisonnable. Ainsi, des problèmes composés de 800 tâches, 20 agents et 700 contraintes ont été envisagés et ont conduit à des MDPs composés, en moyenne par agent, de 57 200 états.

10.2 Algorithme de révision des politiques

La mise en place de l'algorithme de révision des politiques a eu pour principal objectif l'élaboration d'un algorithme de planification multi-agent permettant de traiter des problèmes de taille conséquente tout en fournissant des solutions de « bonne » qualité. Nous allons, dans cette section, décrire différentes expérimentations menées afin de vérifier que l'algorithme développé répond bien à ces exigences. Nous nous intéresserons tout d'abord à l'efficacité de cet algorithme et nous estimerons la taille des problèmes pouvant être résolus. Nous étudierons ensuite la qualité des solutions calculées que nous tenterons de comparer avec les autres approches existantes.

10.2.1 Efficacité de l'algorithme

L'étude de l'évolution du nombre d'états menée dans la section précédente, nous conduit naturellement à nous interroger sur la taille des espaces d'états pouvant être traités par notre algorithme, ainsi que sur le temps nécessaire à la résolution de différentes tailles de problèmes.

Comme le prouvent les corollaires 3 et 4 concernant la complexité temporelle de l'algorithme de révision des politiques, le temps nécessaire afin de réviser l'ensemble des politiques des agents dépend du nombre d'états de chaque agent, ainsi que du nombre d'actions possibles en chaque état.

Evaluation du nombre d'actions

L'étude de la taille des espaces d'états qui vient d'être réalisée nous a conduit à examiner les différents paramètres liés au nombre d'intervalles d'exécution de chaque tâche. Nous avons ainsi montré que les contraintes temporelles, les contraintes de précédence et les durées d'exécution influencent le nombre d'intervalles d'exécution. Ces paramètres influencent également le nombre de dates de début de toute tâche t_i et conditionnent par conséquent le nombre d'actions à envisager lors de la révision des politiques. Le nombre d'actions pouvant être exécutées à partir d'un état s_i dépend, en effet, du nombre de dates de début possibles pour la prochaine tâche. Ainsi, plus les dates de début possibles pour une tâche sont nombreuses, plus le nombre d'actions à envisager est important et plus la résolution du problème requiert de temps.

Considérons une tâche t_i devant être exécutée à partir d'un état s_i. Soit $|ST|$ le nombre maximum de dates de début pour une tâche. Il existe alors, au pire cas, $|ST|$ actions possibles à partir de s_i. Parmi l'ensemble des dates de début de t_i, certaines n'ont néanmoins pas à être

envisagées. Le nombre de dates de début possibles pour l'exécution d'une tâche t_i dépend en effet de l'ensemble des dates de début de t_i mais également des connaissances sur les dates de fin des prédécesseurs de t_i. Considérons deux tâches t_{i-1} et t_i telles que t_{i-1} soit un prédécesseur de t_i. Si t_i et t_{i-1} sont exécutées par un même agent $\mathcal{A}g_i$, la date de fin et_{i-1} de t_{i-1} est alors connue de $\mathcal{A}g_i$ lorsque celui-ci doit décider quand exécuter t_i. Afin de déterminer l'ensemble des dates de début possibles de t_i, il est donc possible de restreindre l'ensemble des dates de début de t_i en éliminant toutes celles qui sont inférieures à et_{i-1}. Supposons à présent que t_{i-1} soit exécutée par un autre agent $\mathcal{A}g_j$. Lorsque $\mathcal{A}g_i$ doit décider quand exécuter t_i, la date de fin de t_{i-1} n'est pas connue, il n'est donc pas possible de restreindre l'ensemble des dates de début de t_i. Toutes les dates de début de t_i sont considérées comme possibles et un plus grand nombre d'actions est alors à envisager.

Ainsi, le nombre d'actions à considérer afin d'établir la politique d'exécution de t_i à partir d'un état s_i dépend des connaissances sur les dates de fin des prédécesseurs. Si aucun des prédécesseurs de t_i n'est exécuté par $\mathcal{A}g_i$, celui-ci ne dispose d'aucune information sur les dates de fin des prédécesseurs et toutes les dates de début de t_i doivent être envisagées. Si un des prédécesseurs de t_i est exécuté par $\mathcal{A}g_i$, il est en revanche possible de réduire l'ensemble des dates de début afin de limiter les actions possibles.

Temps d'exécution

Le tableau 10.1 décrit, pour un même ensemble de tâches, le temps nécessaire à l'exécution de l'algorithme de révision des politiques en fonction du nombre d'agents chargés de réaliser les tâches. Le nombre d'états total à évaluer ainsi que le nombre d'actions considérées au cours de la révision des politiques sont également mentionnés. Les résultats présentés mettent en évidence l'influence du nombre d'états et du nombre d'actions sur le temps de résolution. Nous pouvons de plus constater que le fait d'augmenter le nombre d'agents ne garantit pas une accélération de la planification. L'ajout d'un agent suscite en effet la re-répartition des tâches qui, comme nous l'avons déjà précisé, accroît les dépendances entre les agents. Ceci augmente également, comme nous venons de l'illustrer, l'incertitude sur les dates de fin des prédécesseurs des tâches t_i d'où, une hausse du nombre d'actions à envisager. De ce fait, pour un même ensemble de tâches, plus le nombre d'agents considérés augmente, plus le nombre d'actions possibles en chaque état augmente. Ainsi, bien que le nombre d'états à considérer pour la mission comprenant 10 agents soit plus faible que le nombre d'états de la mission impliquant 2 agents, le nombre d'actions à considérer est beaucoup plus important. L'exécution de l'algorithme de révision des politiques requiert alors plus de temps pour 10 agents que pour 2 agents.

Nombre de tâches	Nombre d'agents	Taille de l'espace d'états	Nombre d'actions à envisager	Temps de résolution (s.)
50	2	73 202	222 696	138
50	3	142 477	153 336	95
50	4	123 189	208 301	154
50	5	95 172	326 907	292
50	10	56 873	702 553	400
50	15	48 697	548 435	350

TAB. 10.1 – Temps d'exécution de l'algorithme de révision des politiques en fonction du nombre d'agents

Bien que de nombreux paramètres influencent la taille des espaces d'états et le nombre d'actions à envisager à partir de chaque état, nous avons tenté d'établir un ordre de grandeur du temps nécessaire à la résolution de différentes tailles de mission. Pour ce faire, un échantillon diversifié de problèmes a été constitué. Nous avons ainsi fait varier la taille des missions, l'importance des contraintes et des resources initiales. La résolution de ces missions nous a ensuite permis d'estimer, en fonction de la taille des problèmes envisagés, le temps moyen nécessaire pour exécuter l'algorithme de révision des politiques.

De manière générale, la résolution d'un problème comprenant moins de 30 000 états requiert moins de 15 secondes[48]. Des missions composées de moins de 150 000 états sont quant à elles résolues en moins de 5 minutes. Enfin, la révision d'un million d'états demande entre une à deux heures suivant le nombre d'actions à considérer.

Le tableau 10.2 résume ces résultats en termes de nombre de tâches et d'agents. Les résultats obtenus montrent que des problèmes composés de plusieurs dizaines de tâches et d'agents peuvent être rapidement résolus. Des missions comme celles rencontrées en robotique exploratoire et consistant en l'exécution d'une centaine de tâches par une dizaine d'agents, peuvent également être aisément considérées. Bien que la planification de l'exécution de plusieurs centaines de tâches nécessite plus de temps, elle reste tout à fait envisageable. Ces expérimentations prouvent ainsi l'efficacité de notre approche. L'algorithme que nous avons développé permet en effet de traiter des problèmes de taille importante en un temps raisonnable. Il offre donc les garanties d'efficacité nécessaires afin de pouvoir être utilisé pour planifier l'exécution de scénarios de la robotique exploratoire tels que ceux présentés au chapitre 4.

[48] Etude réalisée sur un Pentium III, 700MHz

Nombre de tâches	Nombre d'agents	Temps de résolution
20	2	8s.
20	10	8s.
50	2	2 min
50	10	1 min
100	2	15 min
100	10	20 min
200	2	1h30

TAB. 10.2 – Estimation du temps d'exécution de l'algorithme de révision des politiques

Révision décentralisée des politiques

Les expérimentations qui viennent d'être présentées ont été réalisées à l'aide de la version centralisée de l'algorithme de révision des politiques. Le gain en temps suscité par l'exécution décentralisée de cet algorithme est très variable selon les problèmes. Il est en effet largement influencé par le nombre de branchements du graphe de la mission et par la durée des communications entre les agents.

Sur de petits problèmes, ce gain est négligeable par rapport au temps d'exécution de l'algorithme. Lorsque la communication est coûteuse en temps, la version décentralisée de l'algorithme s'avère même moins efficace. Pour des graphes de taille importante (plus de 50 tâches), la version décentralisée de l'algorithme permet un gain en temps, à condition que la topologie du graphe se prête à la révision simultanée des politiques d'exécution de différentes tâches. Il est donc nécessaire que chaque niveau du graphe contienne un grand nombre de nœuds, ce qui n'est possible que pour des missions impliquant un grand nombre d'agents. Le gain en temps suscité par la révision décentralisée des politiques dépend alors du temps de communication et du nombre de valeurs de coût occasionné à communiquer. Pour des missions de taille importante propices à une telle exécution de l'algorithme de révision des politiques, nous avons ainsi pu constater un gain de plusieurs minutes.

10.2.2 Étude qualitative des solutions

Les expériences qui viennent d'être présentées ont permis d'établir l'efficacité de l'algorithme de révision des politiques. Nous sommes alors naturellement amenés à nous interroger sur la qualité des solutions ainsi calculées.

Une étude difficile

L'évaluation de la qualité des solutions retournées par l'algorithme de révision des politiques n'est toutefois pas une tâche aisée. Nous avons en effet évoqué à plusieurs reprises les difficultés, voire l'impossibilité de résoudre de manière optimale les problèmes qui nous sont posés. Nous ne pouvons donc pas envisager mesurer la qualité des solutions que nous calculons en les comparant à la solution optimale.

Par ailleurs, les approches existantes de résolution des DEC-MDPS qui s'orientent vers la recherche d'une solution approchée, ne traitent aucune des contraintes prises en compte par notre approche (contraintes de précédence ou contraintes temporelles). La modélisation et la résolution, par de telles approches, des problèmes considérés n'est donc pas envisageable puisqu'elles supposeraient que les contraintes ne soient pas prises en compte et conduiraient donc à des solutions de moins bonne qualité puisque ne respectant pas les contraintes.

En ce qui concerne les approches de planification autres que celles se basant sur l'utilisation des DEC-MDPs, il nous a également été impossible de réaliser des comparaisons. Ces approches ne permettent en effet pas la représentation de toutes les caractéristiques (incertitude, contraintes temporelles et de ressources,...) des problèmes que nous envisageons.

Comparaison qualitative de différentes politiques

Afin d'étudier la qualité des solutions retournées par l'algorithme de révision des politiques, une comparaison a été menée entre les politiques calculées par notre approche et différentes politiques telles que la politique EST ou la politique LST[49]. Nous nous sommes ainsi intéressée au gain total obtenu par les agents et au nombre d'échecs partiels réalisés. Nous avons pour cela comparé les performances résultant de l'exécution des politiques suivantes :

- la politique calculée par notre algorithme en considérant la politique EST comme politique initiale et en utilisant la mesure de coût occasionné la plus précise (politique EOC)[50],
- la politique EST,
- la politique LST,
- la politique P (Probabiliste) consistant à sélectionner, pour toute tâche t_i, la date de début à laquelle les prédécesseurs de t_i ont la plus forte probabilité de terminer.

Remarquons que le calcul de chacune de ces politiques nécessite plus ou moins de temps. Afin de déterminer les politiques EST et LET, il est uniquement nécessaire d'établir les dates de début de chaque tâche. La politique P requiert, quant à elle, le calcul des probabilités sur les dates de fin des tâches. Enfin, la politique EOC déterminée par notre approche nécessite la

[49] Ces politiques ont été présentées au chapitre 8.

[50] Une comparaison de la qualité des solutions en fonction des mesures du coût occasionné sera détaillée à la fin de ce chapitre.

modélisation du problème sous forme d'OC-DEC-MDP et l'exécution de l'algorithme de révision des politiques.

Bien que le calcul de la politique EOC nécessite plus de temps, notre approche permet d'obtenir des solutions de qualité supérieure. Le tableau 10.3 réalise une comparaison des résultats obtenus au cours de 1000 exécutions d'une mission mettant en jeu deux agents devant réaliser une vingtaine de tâches. Nous pouvons ainsi constater que le gain obtenu par les agents qui suivent la politique EOC est supérieur à celui obtenu lors de l'exécution des autres politiques recensées. Une étude plus détaillée des exécutions a d'ailleurs permis d'observer que la politique EOC permettait, le plus fréquemment, de mener à bien l'exécution de toutes le tâches.

	EST-pol.	LST-pol.	P-pol.	EOC-pol.
Partial failures	519	0	114	113
Gain	8274	9600	9888	10059

TAB. 10.3 – Comparaison des performances de différentes politiques

L'étude du nombre d'échecs partiels nous a permis d'estimer le nombre de fois où l'un des agents décide d'exécuter une tâche alors qu'il aurait pu attendre. Plus les ressources d'un agent sont restreintes, plus les échecs partiels doivent être minimisés. Notre approche permet alors de minimiser les échecs partiels tout en maximisant le gain des agents. Le tableau 10.3 montre que le nombre d'échecs partiels est le plus important lorsque les agents suivent la politiques EST. En effet, chacun décide de commencer l'exécution de la prochaine tâche au plus tôt. La probabilité que le prédécesseurs n'aient pas terminé est alors forte d'où, un nombre élevé d'échecs partiels. Les résultats présentés dans la tableau 10.3 ont été obtenus à partir de l'exécution d'une mission sous ressources restreintes. Le gain des agents suivant la politique EST est donc faible puisque le grand nombre d'échecs partiels provoqués par cette politique conduit rapidement à un échec total par manque de ressources. Le tableau 10.4 décrit le gain des agents pour l'exécution de cette même mission, lorsque les ressources sont illimitées. Nous constatons que la politique EOC et la politique EST permettent toutes deux d'obtenir le gain maximum. Comme nous l'avons démontré au chapitre 8, ces deux politiques sont en effet identiques et optimales.

Les performances de la politique LST s'expliquent quant à elles par la forte probabilité d'échouer totalement par violation des contraintes temporelles. Lorsque les ressources des agents sont restreintes, il est possible que les politiques EOC et LST soient très proches. En effet, si les récompenses des tâches restant à exécuter sont faibles par rapport aux récompenses déjà acquises, la politique EOC des tâches à venir correspondra à la politique LST. Le premier cas de la figure 10.13 décrit une telle situation. Lorsque l'agent 1 vient d'exécuter t_j, les récompenses à

	EST-pol.	LST-pol.	P-pol.	EOC-pol.
Gain	10100	9550	9974	10100

TAB. 10.4 – Comparaison des performances de différentes politiques sous ressources illimitées

venir sont faibles. La pénalité induite par l'échec total de ces tâches est donc également faible. Si l'agent dispose de peu de ressources, il va alors chercher à minimiser les échecs partiels pour économiser ses ressources. Il va donc exécuter chaque tâche au plus tard (politique LST) même s'il risque ainsi de violer les contraintes temporelles : cette politique est la seule lui permettant de mener à bien l'exécution de la mission et les échecs totaux sont peu pénalisants. En revanche, si une forte récompense est attribuée à l'une des tâches t_k restantes, la politique LST va s'avérer trop risquée : la perte provoquée par un échec de t_k étant forte, l'agent va chercher à assurer l'exécution de cette tâche au détriment des tâches suivant t_k. Ainsi, l'agent va adopter, au risque d'échouer partiellement, une politique peu risquée[51] jusqu'à ce que la tâche t_k soit exécutée. Une politique proche de la politique LST sera ensuite exécutée. Considérons le deuxième cas de la figure 10.13, l'agent 1 va adopter une politique non risquée afin de pouvoir exécuter sa dernière tâche. Dans le troisième cas, l'agent 1 va suivre une telle politique jusqu'à l'exécution de t_k. Enfin, dans le quatrième cas l'agent va suivre une politique non risquée jusqu'à l'exécution de t_l. En effet, échouer totalement cette tâche provoquerait un coût occasionné important sur l'agent 2. L'agent 1 va donc chercher à minimiser la violation des contraintes temporelles même si cela implique des échecs partiels qui, par leur consommation de ressources, l'empêcheront d'exécuter toutes les tâches qu'il lui reste.

L'écart entre les politiques P et EOC s'explique par la présence des contraintes temporelles. Lorsque ces dernières ne risquent pas d'être violées par la politique P, la politique EOC peut s'avérer proche[52] de la politique P. Aucun mécanisme de coordination entre les agents n'est toutefois mis en place par la politique P ce qui explique qu'elle conduise à de moins bons résultats. Contrairement à la politique EOC, l'influence d'une action sur les autres agents n'est en effet pas envisagée. La politique EOC garantit donc, à la différence de la politique P, un comportement coopératif de la part des agents menant ainsi à de meilleures performances.

[51] Nous désignerons par le terme « politique risquée », une politique pouvant mener avec une forte probabilité à un échec total. La politique LST est donc une politique risquée.

[52] Nous déterminons la proximité de deux politiques en comparant les décisions prises par chacune d'elles en chaque état.

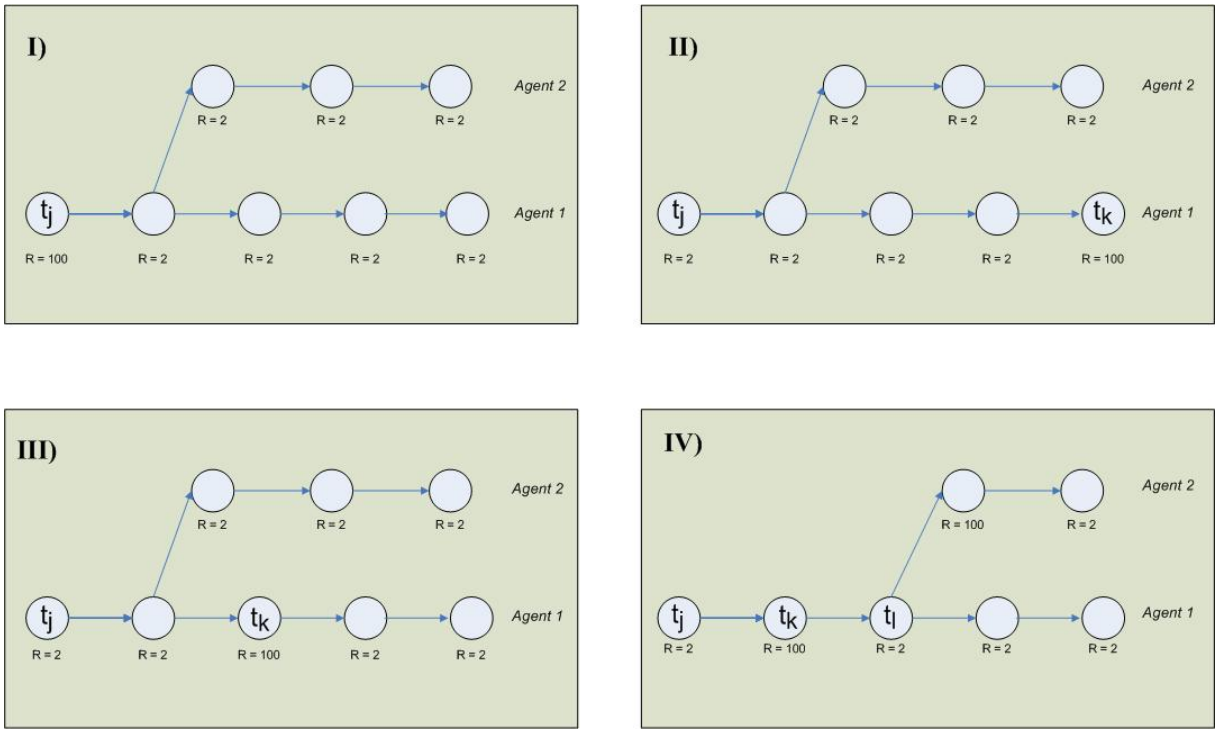


FIG. 10.13 – Influence des récompenses sur les politiques

Suivant les contraintes du problème et les ressources initialement disponibles, la politique EOC se rapproche donc plus ou moins d'une des politiques extrêmes LET ou LST. Les expérimentations ont montré que la politique EOC fournit de meilleurs résultats que les autres politiques qui ont été présentées. Elle permet en effet de décider des actions à exécuter tout en gérant les ressources ansi que les contraintes temporelles et de précédence. Toute décision résulte alors d'un dosage précis entre la probabilité d'échouer partiellement, la consommation d'un tel échec, la probabilité d'échouer totalement, le gain à venir et le coût occasionné sur les autres agents. Plus les ressources des agents sont limitées, plus il est important de limiter les échecs partiels et plus la politique EOC s'éloigne de la politique EST. L'algorithme de révision des politiques permet alors d'établir un juste milieu entre la politique EST et la politique LST, afin d'économiser les ressources tout en limitant les risques de violation des contraintes temporelles.

Quand l'optimal peut être calculé

Bien que la résolution optimale des problèmes qui nous sont posés ne soit de manière générale pas envisageable, l'algorithme CSA développé par Becker et al. [Becker *et al.*, 2004a] et présenté en première partie de ce document, peut être utilisé afin de résoudre des problèmes de petite dimension. Cet algorithme permet de calculer la politique optimale de chaque agent tout en respectant les contraintes de précédence entre les tâches. En pratique, cet algorithme ne peut

résoudre que de très petits problèmes ne dépassant pas deux agents, une dizaine de tâches et quelques contraintes de précédence entre les agents. Lorsque ces dernières sont orientées aussi bien vers le premier agent que vers le deuxième, la résolution du problème par CSA requiert un temps trop important et aucune solution ne peut être obtenue. Une solution optimale ne peut donc être calculée que pour des graphes à sens unique. Nous avons démontré que, pour de tels graphes, l'algorithme de révision des politiques calcule une politique optimale. Nous pouvons donc affirmer que notre approche égalise les performances de CSA en terme de qualité de la solution, sur les problèmes que CSA peut résoudre en pratique.

L'algorithme de révision des politiques que nous avons proposé étant de plus polynomial en espace d'états alors que CSA est exponentiel, une solution est plus rapidement obtenue par notre approche.

Du fait de sa complexité, CSA n'est pas en mesure de calculer la solution optimale aux problèmes de grande taille que nous avons évoqués dans ce chapitre. Pour ces problèmes, CSA ne peut donc pas être utilisé afin de mener à bien l'étude de la qualités des solutions fournies par notre approche.

10.3 Algorithme itératif de révision des politiques

Les expérimentations qui viennent d'être présentées montrent que l'algorithme de révision des politiques que nous avons mis en place est effectivement en mesure de résoudre des problèmes de taille importante. Il répond ainsi aux objectifs que nous nous étions fixés concernant les dimensions des missions à gérer. Nous sommes à présent amenés à nous interroger sur l'efficacité de l'algorithme itératif et à étudier l'impact du processus itératif sur la taille des problèmes pouvant être traités.

10.3.1 Étude de l'efficacité

Comme nous l'avons démontré au chapitre 9, la complexité spatiale de l'algorithme itératif est identique à celle de l'algorithme de révision des politiques. Une même quantité de mémoire étant requise par ces deux algorithmes, tout problème pouvant être résolu par l'algorithme de révision des politiques peut *a priori* être traité par l'algorithme itératif.

Le temps d'exécution de l'algorithme itératif est cependant plus important puisqu'il dépend du nombre d'itérations réalisées. A chaque itération, l'algorithme de mise à jour des fonctions de transition est exécuté puis, la révision des politiques est réalisée. Le temps requis par une itération est donc égal à la somme des durées d'exécution de ces deux algorithmes. Le temps nécessaire afin de mettre à jour les fonctions de transition est toutefois négligeable par rapport à la durée d'exécution de l'algorithme de révision des politiques. Nous avons en effet comparé, pour différentes tailles de mission, les durées d'exécution de ces deux algorithmes. Nous avons ainsi

pu constater que la mise à jour des fonctions de transition s'effectuait très rapidement : sur des problèmes d'une vingtaine de tâches, elle ne nécessite que 5 à 6 millisecondes, elle requiert 3 à 4 secondes sur des problèmes d'une centaine de tâches, et 5 à 6 secondes pour des missions de 200 tâches. Le temps nécessaire par une itération est ainsi majoritairement conditionné par la durée de révision des politiques et non par la mise à jour des fonctions de transition. Les 5 à 6 millisecondes nécessaires pour calculer les probabilités de transition sont en effet négligeables par rapport aux 8 secondes requises pour réviser les politiques. Ceci confirme les résultats obtenus par l'étude de la complexité de l'algorithme itératif réalisée au chapitre 9. Nous avions alors démontré que la complexité d'une itération était du même ordre que la complexité de l'algorithme de révision des politiques.

L'évolution de la durée d'une itération, en fonction de la taille des missions considérées, est donc analogue à l'évolution des durées d'exécution de l'algorithme itératif. Il est possible de se référer à l'étude précédemment menée pour de plus amples détails.

10.3.2 Rapidité de convergence

Étant donnée l'influence du nombre d'itérations sur le temps d'exécution de l'algorithme itératif, nous avons mené différentes expérimentations permettant de mieux apprécier cette donnée. Une grande diversité de missions a ainsi été résolue par l'algorithme itératif et le nombre d'itérations nécessaires pour converger a été étudié. Nous avons également fait varier les méthodes d'estimation du coût occasionné afin d'évaluer leur influence sur le processus de convergence.

Sur l'ensemble des expériences qui ont été menées, nous n'avons jamais été confrontée à un cas de divergence de l'algorithme. Bien que la convergence ne soit en théorie pas garantie, il semble donc qu'en pratique les situations de divergence soient très rares. Dans la majeure partie des cas, l'équilibre correspondant à la convergence de l'algorithme a de plus été atteint en moins de quatre itérations.

Les expérimentations qui ont été réalisées ont mis en évidence l'influence des ressources initiales sur le nombre d'itérations. Lorsque les ressources sont illimitées ou en quantité importante, seule une itération est nécessaire pour atteindre l'équilibre. La politique EST correspondant à la politique optimale est alors obtenue à l'issue de cette itération. De la même façon, une très faible quantité initiale de ressources se traduit par un petit nombre d'itérations. En effet, si un agent ne dispose pas des ressources suffisantes pour exécuter sa première tâche, sa politique optimale consiste à commencer l'exécution de la première tâche au plus tôt et ce quelle que soit la politique des autres agents. La politique des tâches suivantes est quant à elle indifférente. Entre ces deux extrêmes, le nombre d'itérations augmente puis décroît. La figure 10.14 représente une telle évolution. Le maximum de cette courbe est atteint pour un taux initial de ressources critique permettant tout juste aux agents d'exécuter leurs tâches sans pouvoir échouer partiellement. Le

nombre d'échecs partiels devant être restreint, atteindre l'équilibre nécessite alors de nombreux ajustements d'où le nombre élevé d'itérations.

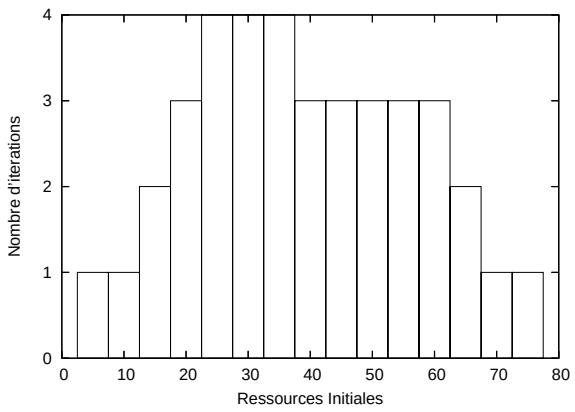


FIG. 10.14 – Influnece des ressources initiales sur le nombre d'itérations

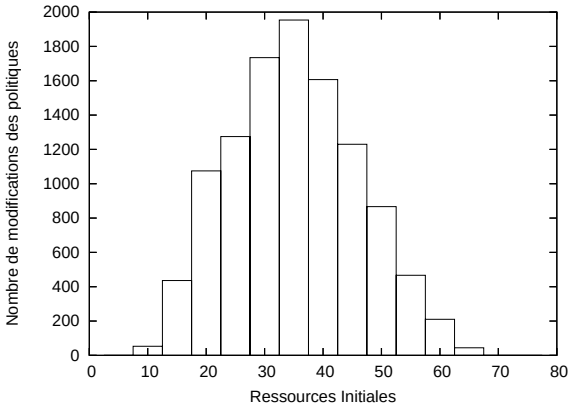


FIG. 10.15 – Évolution des ressources initiales sur le nombre de modifications des politiques

Cette tendance est également identifiable lorsque nous étudions le nombre de modifications des politiques réalisées à chaque itération. La figure 10.15 décrit l'évolution du nombre de modifications en fonction des ressources disponibles au début de la mission. Chaque agent suit initialement la politique EST. Ainsi, lorsque les ressources sont très faibles ou bien en quantité importante, aucune modification n'est réalisée sur les politiques des agents. La politique initiale correspond en effet à un équilibre et plus particulièrement à une solution optimale. En revanche, si les ressources sont tout juste suffisantes (les agents ne peuvent se permettre d'échouer partiellement s'ils souhaitent exécuter toutes leurs tâches) un grand nombre de modifications et d'ajustements sont nécessaires pour atteindre l'équilibre.

Ces résultats mettent en évidence l'influence des politiques initiales sur le nombre d'itérations. Ils suggèrent également la possibilité d'établir le choix des politiques initiales en fonction des ressources disponibles par les agents. Ainsi, si les ressources sont très faibles ou en quantité importante, nous préférerons la politique EST. En revanche, si les ressources sont suffisantes sans plus, une politique probabiliste (politique P) peut s'avérer plus adaptée. Notons toutefois que la preuve de convergence que nous avons établie n'est valable que si les agents suivent initialement la politique EST.

10.4 Comparaison des performances

La mise en place de l'algorithme itératif a eu pour objectif l'augmentation de la qualité des politiques calculées, dans les cas où l'optimalité des solutions retournées par une seule exécution de l'algorithme de révision des politiques n'est pas garantie. Différentes expérimentations ont par conséquent été menées afin d'étudier l'évolution de la qualité des solutions au fur et à mesure de l'exécution du processus itératif. Nous nous sommes en particulier intéressée au gain suscité par chaque itération.

Comme nous l'avons démontré au chapitre 9, la mesure du coût occasionné utilisée afin d'établir les politiques des agents conditionne à la fois les garanties de convergence de l'algorithme itératif et la qualité des solutions obtenues. Nous nous sommes donc également intéressée à l'influence des mesures du coût occasionné sur le processus itératif. Les conséquences de l'utilisation de l'une ou l'autre des mesures sur le nombre d'itérations et sur la qualité des solutions ont ainsi été examinées.

10.4.1 Convergence et mesures du coût occasionné

L'influence, sur le nombre d'itérations, des différentes mesures de coût occasionné a tout d'abord été étudiée. Nous avons fait varier les méthodes d'estimation du coût occasionné afin d'évaluer leur impact sur le processus de convergence. Nous avons alors pu constater que l'utilisation du coût occasionné avec ressources (mesure précise) conduit à un plus grand nombre d'itérations. Ce supplément d'itérations s'explique par la précision de la mesure qui suscite de nombreux ajustements des politiques en réponse aux modifications réalisées aux itérations précédentes. En effet, la mesure du coût occasionné étant précise, elle s'avère sensible aux modifications des politiques et varie à la moindre révision, entraînant ainsi d'autres modifications. De nombreux ajustements sont alors nécessaires afin d'atteindre un équilibre.

En revanche, l'utilisation d'une mesure approchée (par exemple sans tenir compte des ressources) du coût occasionné, restreint les modifications et permet d'atteindre un équilibre plus rapidement. L'approximation du coût occasionné est en effet moins sensible aux modifications des politiques. Une approximation ne tenant pas compte des ressources ne sera pas influencée par une modification des disponibilités des ressources. Toute modification de la politique d'un agent $\mathcal{A}g_i$ entraînant une augmentation ou une diminution des ressources disponibles avant l'exécution d'une tâche t_i ne suscitera pas de changement du coût occasionné de t_i. Les agents $\mathcal{A}g_j$ $(j \neq i)$ tenant compte du coût occasionné sur t_i ne percevront pas ces modifications de la politique de $\mathcal{A}g_i$ et ne réviseront pas leurs politiques en conséquence. Les modifications des politiques sont ainsi moins nombreuses et l'équilibre est atteint plus rapidement. Comme nous le montrerons par la suite, les solutions obtenues pourront toutefois s'avérer de moins bonne qualité.

Des expérimentations ont été menées afin d'estimer le nombre d'itérations réalisées pour

les différentes méthodes de coût occasionné. Celui-ci varie bien entendu selon les problèmes. Le nombre d'itérations effectuées en cas d'utilisation du coût occasionné espéré avec ressources dépassant rarement les quatre itérations. En moyenne, le nombre d'itérations réalisées, lorsque la mesure du coût occasionné espéré avec ressources est utilisée, est légèrement supérieur à 3 itérations. Il est en revanche proche de 2 itérations lorsque la mesure du coût occasionné sans ressources est utilisée.

Le gain en temps dépend alors de la durée d'une itération. Faire l'économie d'une itération est en effet beaucoup plus intéressant sur de grandes missions. Pour de tels problèmes, il est donc préférable d'avoir recours à des méthodes d'approximation du coût occasionné. Lorsque des problèmes de petite taille sont considérés, nous pouvons à l'inverse nous permettre une mesure plus fine du coût occasionné au risque d'augmenter le nombre d'itérations.

10.4.2 Profil d'amélioration

L'évolution, à chaque itération, de la qualité des solutions a par ailleurs été étudiée. Conformément à ce qui a été démontré au chapitre 9, l'estimation du coût occasionné sans ressources ou à l'aide d'une méthode mixte conduit à une augmentation de la qualité de la solution à chaque itération. Un tel profil d'amélioration est également repérable lorsque le coût occasionné est précisément estimé (coût occasionné espéré avec ressources). Il semble en effet que la baisse des performances des agents soit, malgré l'utilisation de cette mesure, un phénomène très rare. Ceci est bien entendu à mettre en relation avec le fait que nous n'ayons rencontré aucun cas de divergence bien qu'une grande diversité de problèmes ait été traitée. Toute divergence émane en effet d'une baisse des performances des agents.

Quelle que soit la méthode d'estimation du coût occasionné utilisée, nous avons pu constater que la première itération de l'algorithme apportait le plus grand gain de qualité. C'est en effet au cours de cette première itération que les performances des agents augmentent le plus. Nous pouvons en déduire que l'algorithme de révision des politiques, quand il n'est exécuté qu'une seule fois comme cela a été présenté au chapitre 8, conduit à de bons résultats. Itérer cette révision des politiques permet alors de réaliser des ajustements des politiques conduisant à une diminution du nombre d'échecs partiels. La figure 10.16 présente l'évolution du nombre de modifications des politiques des agents à chaque itération, pour une mission comprenant deux agents et vingt tâches. La première itération réalise le plus grand nombre de modifications. Ce dernier décroît ensuite pour devenir nul à la troisième itération. Ainsi, deux itérations sont nécessaires pour atteindre l'équilibre.

La figure 10.17 décrit, pour le même problème, le nombre d'échecs partiels réalisés par les agents lorsque les politiques résultant d'une itération donnée sont exécutées. Le nombre d'échecs partiels est comptabilisé sur 1000 exécutions et l'itération 0 correspond à l'ensemble des politiques

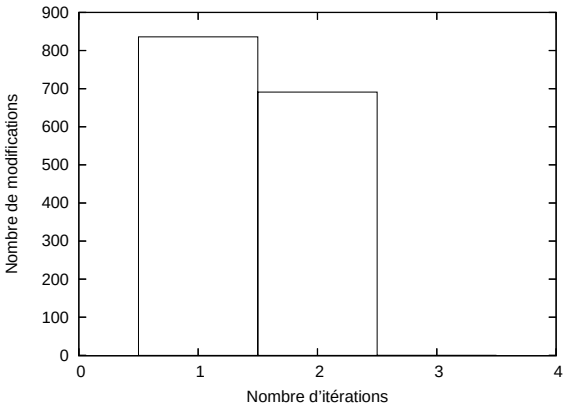


FIG. 10.16 – Nombre de modifications des politiques à chaque itération

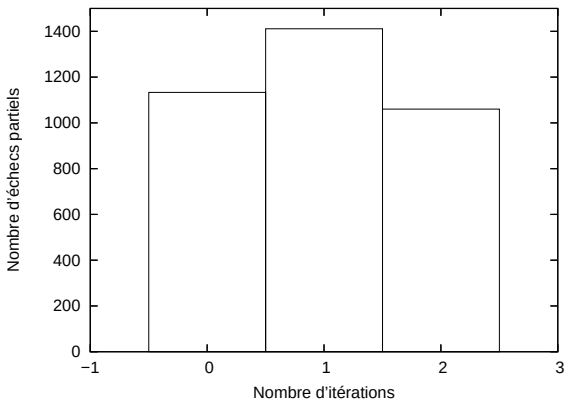


FIG. 10.17 – Nombre d'échecs partiels à chaque itération

initialement fixées (ici la politique EST). Bien que l'ensemble des politiques résultant de la première itération soit presque optimal, il conduit néanmoins à un grand nombre d'échecs partiels. Ces derniers sont essentiellement dûs au fait que la politique d'exécution de chaque tâche n'est révisée qu'une seule fois. La politique d'exécution d'une tâche t_i ne peut donc être ajustée en fonction des modifications des politiques d'exécution des tâches en amont de t_i dans le graphe de la mission. La seconde itération permet ces ajustements d'où une diminution du nombre d'échecs partiels et une augmentation des performances.

Lorsque les ressources initiales des agents sont tout juste suffisantes, un plus grand nombre d'itérations est nécessaire afin de converger. En raison du manque de ressources, toute modification de la politique d'un agent $\mathcal{A}g_i$ a en effet des répercussions beaucoup plus importantes sur les autres agents. De nombreux ajustements sont alors nécessaires afin d'atteindre un équilibre.

Bien que l'augmentation des performances au fil des itérations soit une constante, même pour la mesure du coût occasionné la plus précise, nous pouvons noter des variations de la qualité des solutions en fonction des méthodes utilisées pour l'estimation du coût occasionné. La figure 10.18 décrit l'influence de ces méthodes sur le gain obtenu par les agents. Ces résultats ont été obtenus à partir de 1000 exécutions des politiques résultant de la première itération et différents taux de ressources initiaux ont été envisagés. Naturellement, plus les agents disposent initialement de ressources, plus leur gain est important puisque la probabilité de manquer de ressources diminue. Lorsque les ressources initiales sont importantes ou très faibles, la mesure de coût occasionné utilisée a peu d'impact sur les performances des agents. En effet, les ressources influencent alors peu le coût occasionné et des politiques identiques sont calculées quelle que soit

la méthode d'estimation à laquelle il est fait appel. Les résultats présentés sur la figure 10.18 suggèrent que, dans les cas où les ressources sont tout juste suffisantes, le coût occasionné espéré avec ressources (mesure précise) conduit, à la première itération, à de meilleures performances. Cette conjecture n'est pas toujours vérifiée. La figure 10.19 décrit l'évolution des performances des agents au fil des itérations étant données différentes mesures du coût occasionné. Dans le cas qui est présenté, nous pouvons constater que l'approximation du coût occasionné permet, dès la première itération, d'obtenir une solution de meilleure qualité que celle calculée en utilisant la mesure précise. L'utilisation du coût occasionné espéré avec ressources nécessite alors un plus grand nombre d'itérations afin d'atteindre une solution de qualité équivalente. L'amélioration de cette solution est toutefois beaucoup moins limitée par la suite. L'estimation précise du coût occasionné permet de continuer à améliorer la qualité de la solution et ainsi d'obtenir de meilleures performances. L'utilisation d'une approximation du coût occasionné ne permet pas, quant à elle, une amélioration des politiques obtenues à la première itération.

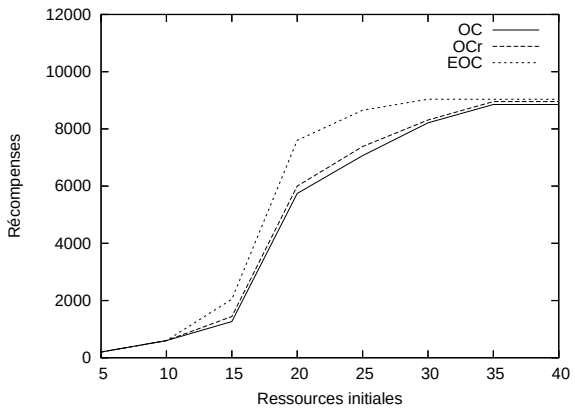


FIG. 10.18 – Influence de la mesure de coût occasionné sur les performances

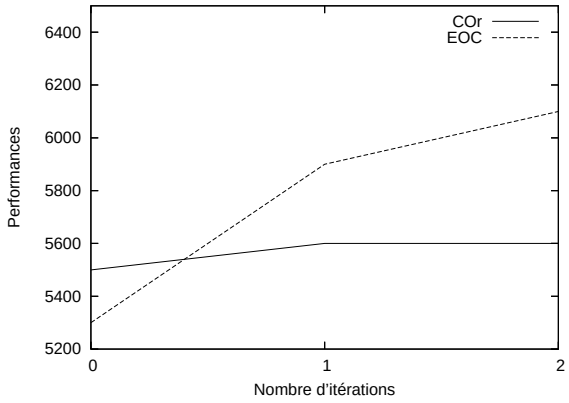


FIG. 10.19 – Evolution des performances en fonction du nombre d'itérations et de la mesure du coût occasionné

Les différentes mesures de coût occasionné conduisent donc à des évolutions différentes des qualités des politiques et nécessitent un plus ou moins grand nombre d'itérations pour converger. Bien que les expérimentations aient montré qu'un équilibre était généralement plus rapidement atteint en utilisant une approximation du coût occasionné, rien ne garantit qu'une telle mesure converge toujours plus rapidement. Le nombre de révisions nécessaires pour atteindre un équilibre peut en effet être plus important en ayant recours à une approximation du coût occasionné.

Il est par ailleurs difficile de comparer les résultats obtenus, à la $k^{\text{ème}}$ itération, pour chacune

des mesures du coût occasionné. Les politiques évoluant dans des directions et à des vitesses différentes, il est préférable de comparer les politiques obtenues après convergence. Dans la majeure partie des cas (85%), l'utilisation du coût occasionné espéré avec ressources nous a permis d'obtenir des solutions de meilleure qualité que celles obtenues en ayant recours à d'autres méthodes d'estimation du coût occasionné. Rien ne permet toutefois de garantir que l'estimation précise du coût occasionné mène toujours à de meilleures performances. L'approximation du coût occasionné peut en effet nous amener à un équilibre de meilleure qualité que celui obtenu par la mesure précise du coût occasionné. Enfin, les différences de qualité sont plus particulièrement sensibles lorsque les ressources des agents sont tout juste suffisantes. Lorsque les ressources sont illimitées ou insuffisantes, le choix de la mesure de coût occasionné a peu d'influence et les solutions calculées sont équivalentes.

10.4.3 Comparaison avec l'existant

L'étude comparative des performances de l'algorithme itératif avec celles d'autres approches existantes est, comme pour l'algorithme de révision des politiques, difficile. Parmi les problèmes que nous traitons, certains peuvent être résolus de manière optimale par CSA. En effet, CSA est capable de résoudre, en pratique, des problèmes représentés par des graphes à sens unique. Pour ces problèmes, il a été précédemment démontré que nos algorithmes calculent également la solution optimale. Notre approche égalise ainsi les performances de CSA en terme de qualité de la solution. Une itération étant de complexité polynomiale alors que CSA est de complexité exponentielle, notre algorithme permet de plus de déterminer plus rapidement la solution optimale. CSA permet néanmoins de résoudre des problèmes autres que ceux que nous traitons. Cette plus grande généricité des problèmes traités explique en grande partie la complexité exponentielle de l'algorithme.

Comme nous le suggérions en première partie de ce document, l'exploitation des caractéristiques du problème et la recherche d'un solution approchée ont permis d'augmenter de façon significative la taille des problèmes traités. La prise en compte des contraintes sur l'exécution des missions a en effet contribué au développement d'algorithmes efficaces. Les contraintes de précédence ont tout d'abord facilité la décomposition du problème en permettant l'identification des interactions entre agents. Les contraintes sur l'exécution des tâches ont ensuite permis de diminuer la tailles des MDPs à résoudre (limitation des espaces d'états) et restreindre l'ensemble des politiques à envisager. Nous rejoignons ici les idées exposées par Nair et al. [Nair *et al.*, 2005] afin de développer une approche efficace pour la résolution des ND-POMDPs. Ils suggèrent en effet d'exploiter les caractéristiques du problème, dans ce cas la localité des interactions, afin de développer un algorithme calculant une politique approchée de la solution optimale.

Conclusion

Nous avons, dans ce chapitre, expérimenté l'efficacité de notre approche. Nous nous sommes en particulier intéressée au temps nécessaire pour résoudre différentes tailles de mission ainsi qu'à la qualité des solutions obtenues.

La complexité temporelle des algorithmes qui ont été précédemment développés dépendant du nombre d'états du modèle et du nombre d'actions à envisager. Nous avons donc tout d'abord étudié l'influence des paramètres d'une mission (nombre de tâches, d'agents, etc.) sur ces données. Nous avons alors exposé les difficultés liées à l'estimation *a priori* du nombre d'états et d'actions.

Nous nous sommes ensuite intéressée à la durée d'exécution de l'algorithme de révision des politiques. Nous avons alors mis en relation le temps d'exécution de nos algorithmes avec le nombre d'états et d'actions à envisager. Nous en avons déduit une estimation du temps nécessaire pour résoudre différentes tailles de problèmes. Il a ainsi été prouvé que l'algorithme de révision des politiques permettait de traiter des problèmes composés de plus d'une centaine de tâches et d'une dizaine d'agents, autorisant ainsi la planification de l'exécution de missions telles que celles rencontrées en robotique exploratoire. Nous avons alors étudié la qualité des solutions obtenues par cet algorithme. Bien que nous manquions de points de comparaison, en particulier avec l'optimal, les expérimentations ont montré que les politiques obtenues conduisent à de bonnes performances des agents. Elles permettent en effet l'obtention d'un comportement coopératif où toute décision résulte d'un équilibre entre le gain immédiat, la probabilité d'échec partiel, la consommation de ressources d'un tel échec, le gain à venir et le coût occasionné sur les autres agents.

Notre étude a ensuite porté sur l'algorithme itératif. Nous avons ainsi montré que la durée d'une itération était équivalente au temps requis par l'exécution de l'algorithme de révision des politiques. Le nombre d'itérations nécessaire pour atteindre l'équilibre étant généralement faible, l'utilisation de l'algorithme itératif reste tout à fait envisageable afin de résoudre les problèmes qui nous sont posés. Le recours à une approximation du coût occasionné peut toutefois s'avérer préférable pour des problèmes dépassant les 200 tâches, cette estimation réduisant en moyenne le nombre d'itérations. En ce qui concerne la qualité des solutions, nous avons pu constater que les politiques résultant du processus itératif conduisent à de meilleures performances que celles obtenues après une exécution de l'algorithme de révision des politiques. Par ailleurs, la qualité des solutions calculées par le biais d'une mesure précise du coût occasionné dépasse le plus souvent celle des solutions obtenues par approximation du coût occasionné. En revanche, l'utilisation d'une mesure précise nécessite plus d'itérations pour converger et risque, dans certains cas, d'amener à diverger. Nous nous retrouvons ainsi à nouveau face à un compromis entre temps de résolution et qualité des solutions obtenues.

Chapitre 11

Application à un scénario multi-robot

Après avoir testé l'efficacité de notre approche et la qualité des solutions obtenues, nous allons nous intéresser à l'applicabilité de ce travail. Nous avons souligné, au chapitre 4, les difficultés liées à l'utilisation des approches déjà existantes pour la planification de tâches en robotique collective. Nous avons montré que ces travaux ne permettent pas la résolution de problèmes de taille importante et omettent de modéliser un certain nombre de contraintes concernant l'exécution des tâches. Leur utilisation pour la résolution de problèmes réels reste donc limitée.

Les capacités de notre approche à résoudre des problèmes de grande taille et à gérer différents types de contraintes ont été précédemment mises en évidence. Nous avons ainsi démontré que notre approche présentait un certain nombre des qualités requises en vue d'être utilisées pour la planification de problèmes réels. Nous allons à présent nous intéresser aux potentialités d'application de ce travail.

A partir d'un scénario inspiré de la robotique exploratoire, nous montrerons comment formaliser le problème qui nous est posé afin qu'il puisse être résolu par notre approche. En utilisant les politiques qui auront été calculées, nous procéderons ensuite à des tests en simulation ainsi qu'à des exécutions de la mission par des robots physiques. Ces expérimentations nous permettront ainsi de montrer les conditions d'application de nos travaux en robotique collective.

11.1 Description et planification de la mission

Le travail que nous avons réalisé a été à l'origine motivé par le développement de colonies de robots autonomes. Nous avons ainsi cherché à mettre en place une approche se basant sur les processus décisionnels de Markov décentralisés et permettant de planifier l'exécution des tâches d'un ensemble de robots. A la suite de la réalisation des travaux qui viennent d'être présentés, nous avons souhaité tester leur réelle applicabilité à des scénarios de la robotique tels que ceux exposés lors de la présentation du cadre applicatif de notre thèse.

11.1.1 Scénario envisagé

Nous nous sommes pour ce faire inspirée d'un scénario de la robotique exploratoire visant à résoudre un problème de planification similaire à ceux rencontrés par les robots envoyés sur Mars. Nous avons ainsi considéré un ensemble de deux robots devant visiter un certain nombre de sites afin d'y réaliser des photographies et/ou des analyses du sol. L'un des avantages lié au développement de colonies de robots autonomes est de permettre la spécialisation des facultés de chacun. Le premier robot considéré est ainsi spécialisé dans la prise de photos alors que le second est équipé d'une foreuse lui permettant de prélever des échantillons du sol. Il dispose également des appareils de mesure nécessaires afin d'analyser ces prélèvements.

Les robots doivent ainsi explorer un ensemble de huit sites pour lesquels il est nécessaire de prendre une photo, d'analyser le sol ou bien de réaliser ces deux tâches. Les prélèvements risquant néanmoins de modifier la topologie des sites, il est nécessaire de les réaliser après la prise de photos. De plus, lorsque le premier robot réalise les photographies, il est impératif qu'aucun autre robot ne soit présent sur le site afin de ne pas risquer de masquer une partie de l'image. A partir d'une zone de départ donnée, le robot 1 doit alors prendre des photos des sites A, B, D, E, F, H et J. Le robot 2 doit quant à lui réaliser des analyses sur les sites C, D, F, H et I. Les sites considérés par un même robot sont ordonnés de sorte à minimiser les déplacements et par conséquent les consommations de ressources. La figure 11.1 présente le déroulement de la mission. Chacun des robots devant visiter les sites D, F et H, des contraintes de précédence doivent être établies entre les robots. Ainsi le robot 2 peut pénétrer sur le site D si et seulement si le robot 1 l'a quitté. Il en va de même pour les sites F et H.

Étant donné que des expérimentations ont déjà été menées afin de tester la taille des missions pouvant être envisagées, nous nous sommes ici limitée à un scénario comprenant seulement deux robots. Une telle restriction a également été conditionnée par le nombre de robots physiques à notre disposition pour réaliser ces expérimentations. Des scénarios de taille plus importante ont toutefois pu être envisagés en simulation.

De même, nous avons restreint le nombre de sites à visiter afin de limiter à quelques minutes la durée d'observation du déroulement de la mission. Ceci permet par ailleurs de réaliser plusieurs exécutions de la mission sans avoir à recharger les batteries des robots.

Les tâches à réaliser dans cette mission consistent à se déplacer, à prendre des photos ou bien à effectuer des analyses du sol. En raison de la méconnaissance de l'environnement, les durées d'exécution ainsi que les consommations de ressources de ces tâches sont incertaines. Le temps et les ressources nécessaires pour se déplacer d'un site A à un site B dépendent en effet de l'inclinaison du terrain, de la pression des roues du robot, de la rugosité du sol, ... L'analyse

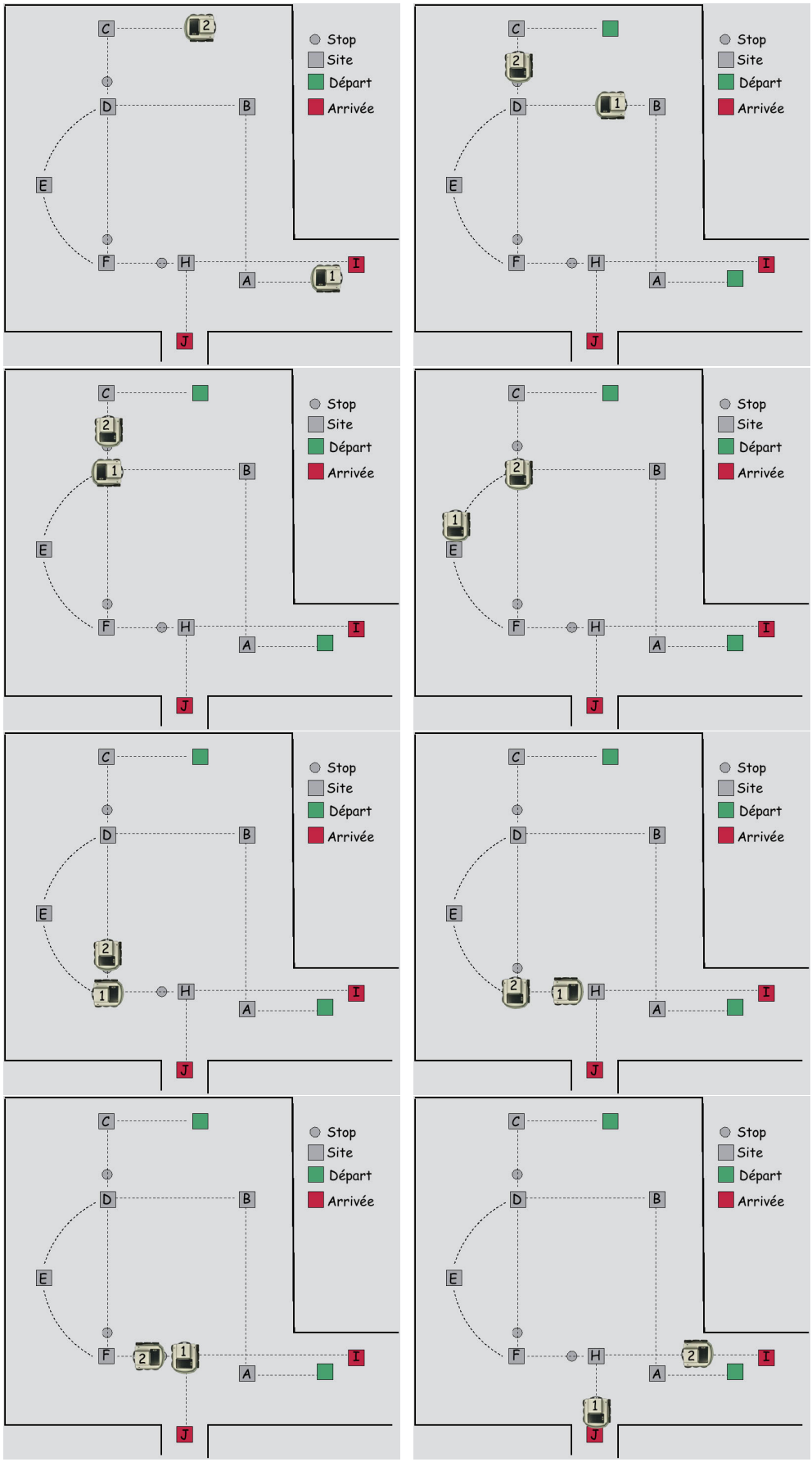


FIG. 11.1 – Scénario envisagé

du sol nécessite quant à elle de prélever un échantillon plus ou moins en profondeur, ce qui peut demander plus ou moins de temps et de ressources suivant la composition du sol. Enfin, la prise de photos requiert des ajustements plus ou moins importants afin d'établir l'angle de prise de vue adéquat. En fonction des conditions d'illumination des sites, des contraintes temporelles sont par ailleurs imposées sur la réalisation des tâches.

Pour qu'une tâche soit exécutée avec succès, les contraintes temporelles et de précédence doivent donc être respectées et le robot doit disposer d'assez de ressources. Chaque fois qu'un robot réalise une tâche t_i avec succès, il reçoit la récompense associée à t_i. L'objectif des robots est alors de maximiser la somme des récompenses qu'ils ont obtenues.

Afin de se coordonner les robots ne peuvent pas communiquer en envoyant des messages. Chacun doit donc anticiper le comportement de l'autre. Ainsi lorsque le robot 2 envisage d'entrer sur le site D, il ne peut demander à l'autre robot s'il l'a quitté. Lorsqu'il tente d'entrer sur le site D, il risque ainsi d'échouer partiellement si le robot 1 est toujours présent sur le site. Le robot 2 réalisera qu'il échoue partiellement au moment où il essaiera d'entrer sur le site, ses capteurs lui indiqueront alors la présence du robot 1 et il sera contraint de ressortir du site. La communication directe n'étant pas possible, la détection des échecs partiels se traduit par une communication indirecte via l'observation de l'environnement.

11.1.2 Planification de l'exécution

Comme le montre la figure 11.2, le scénario envisagé peut être représenté par un graphe orienté acyclique. Chaque nœud représente ainsi une tâche à laquelle sont associées les données qui la caractérisent : durées d'exécution et leurs probabilités, consommations de ressources et leurs probabilités, date de début au plus tôt, date de fin au plus tard, ensemble des prédécesseurs et récompense. Toutes ces données sont déduites de la description du problème ainsi que des connaissances sur l'environnement et sur les robots. Ainsi, les durées d'exécution des tâches et leurs consommations de ressources sont estimées à partir des expériences déjà menées sur les robots dans le même environnement ou dans un environnement similaire. Les contraintes temporelles et de précédence émanent, quant à elles, de la description du problème. Lorsqu'aucune restriction n'est imposée sur la date de début au plus tôt d'une tâche t_i, cette dernière est égale à la date de début de la mission. Si aucune limite n'est définie sur la date de fin de t_i, la date de début au plus tard de t_i est considérée comme infiniment grande.

Notre approche permet ainsi de tenir compte d'un certain nombre de caractéristiques des scénarios de la robotique exploratoire. Les incertitudes sur les durées d'exécution et les consommations de ressources sont prises en compte. Nous considérons également les contraintes temporelles, de précédence et de ressources. Comme nous l'avons mentionné précédemment, d'autres types de contraintes peuvent toutefois intervenir sur le problème. Elles ne sont alors pas prises en

compte par notre approche. Cette dernière permet néanmoins d'envisager les contraintes les plus fréquemment rencontrées dans les applications que nous traitons. Il est par ailleurs difficilement envisageable de répertorier et traiter tous les types de contraintes imaginables dans de telles applications.

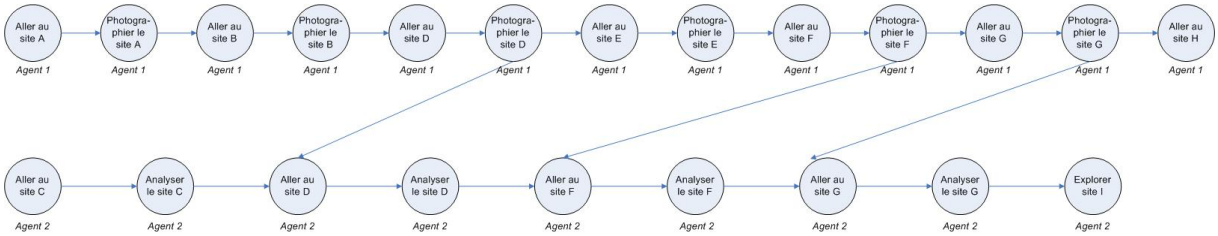


FIG. 11.2 – Graphe de la mission du scénario envisagé

Pour chaque tâche de la mission, nous pouvons renseigner les données la caractérisant comme le montre la figure 11.3 pour la tâche « Prendre une photo du site D ». A partir du graphe ainsi défini, la modélisation du problème sous forme d'OC-DEC-MDP est automatiquement générée, puis le calcul des politiques est réalisé.

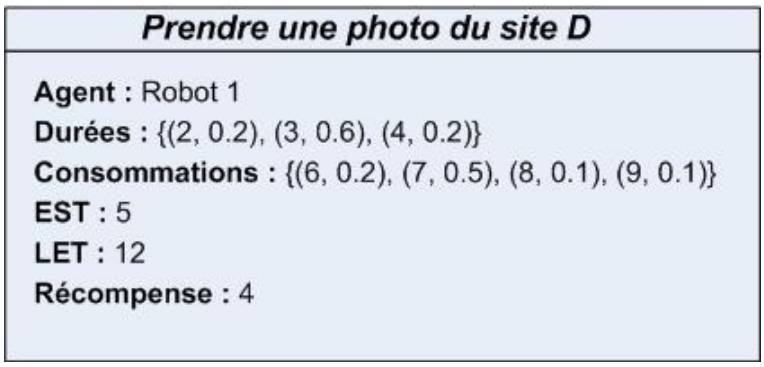


FIG. 11.3 – Détail des données relatives à une tâche

11.2 Exécution d'une mission

Dans le cadre de l'exécution du scénario précédemment décrit, l'élaboration du modèle ainsi que la planification des tâches ont été réalisées sur une machine de bureau standard. Nous avons ainsi obtenu deux politiques (une pour chaque robot) associant une action à chacun des états possibles d'un robot. Les politiques ainsi calculées ont alors été exécutées sur un simulateur et sur des robots physiques[53].

[53] Nous utilisons le terme « robots physiques » par opposition aux robots virtuels utilisés sur les simulateurs.

11.2.1 En simulation

L'exécution de la mission précédemment décrite a tout d'abord été réalisée sur un simulateur. Nous avons ainsi pu étudier le comportement des agents (robots), envisager tous les cas d'exécution possibles et vérifier que les contraintes étaient effectivement respectées.

La réalisation des tâches en simulation nous a permis de nous abstraire des difficultés d'exécution liées à l'imprécision des capteurs et des effecteurs des robots. Les politiques ont ainsi été exécutées dans des conditions idéales : aucune déviation n'émanant des déplacements des robots, aucun bruit ne faussant les mesures des capteurs, etc. La cohérence du comportement des robots a alors pu être vérifiée. Nous avons pu constater que quelles que soient les consommations de ressources et les durées d'exécution des tâches, les politiques des agents conduisaient effectivement au respect des contraintes temporelles, de précédence et de ressources.

Les politiques étant préalablement calculées, l'exécution de la mission se résume au déroulement d'un automate à états finis dans lequel chaque état correspond à un état de l'agent. Le passage d'un état à un autre est alors déterminé par l'action exécutée, sa consommation de ressources et sa durée. Les seules informations devant être connues par un robot sont sa politique et son état courant. Le programme réalisant la prise de décision répond donc aux exigences de légèreté et de simplicité du processus de décision utilisé lors de l'exécution (cf. chapitre 4). Nous pouvons ainsi envisager l'exécution des politiques par des robots physiques ayant des capacités de calcul limitées.

11.2.2 Sur des robots Koala

Le scénario décrit en première section a ensuite été exécuté par les robots Koala présentés en annexe B. L'exécution des politiques par ces robots a nécessité le développement d'un interpréteur de politiques (automate) pouvant être exécuté sur les robots, ainsi que la définition des commandes élémentaires nécessaires à la réalisation de chaque tâche.

Les travaux que nous avons développés ne s'occupent en effet pas de la manière dont sont réalisées les tâches. Nous ne gérons par exemple pas la façon dont le déplacement d'un site A à un site B est effectué. De même, les différentes procédures nécessaires à l'accomplissement des analyses du sol, ne sont pas envisagées. Cette approche permet de rester très générique et de calculer des politiques pouvant s'exécuter sur différents types de robots. La manière dont sont réalisées les tâches dépend ensuite des robots engagés dans la mission. Il est alors nécessaire de définir, en fonction des robots, les commandes élémentaires permettant la réalisation des tâches. En ce qui concerne l'exécution du scénario précédent, les robots Koala que nous avons utilisés ne disposent pas des effecteurs nécessaires à la prise de photos et à l'analyse du sol. L'exécution de ces tâches a donc été concrétisée par une phase d'attente de durée incertaine. Les déplacements ont quant à eux été décomposés en un ensemble de rotations et de mouvements rectilignes en

avant. Les coordonnées de chaque site ont, pour ce faire, été fournies aux robots.

Le passage du simulateur à des robots physiques nous a confrontée au problème d'incertitude des capteurs et des effecteurs. Un calibrage des robots a été nécessaire afin de mettre en place une procédure de correction des déviations lors des déplacements. Cette procédure, adaptée à l'environnement dans lequel nous avons travaillé, nous a ainsi permis de réaliser des mouvements d'une précision suffisante, sans devoir mettre en place un processus de localisation et de correction des trajectoires qui soit trop lourd. Cette procédure peut néanmoins s'avérer inefficace dans un environnement différent de celui que nous avons envisagé.

Les robots utilisés n'étant pas équipés de moyen de communication direct, l'envoi de messages est, comme nous l'avons précédemment supposé, impossible. La vérification du respect des contraintes de précédence a alors été réalisée par l'intermédiaire des capteurs infrarouges. Lorsque le robot 2 désire entrer sur un site, ses capteurs lui permettent de détecter la présence d'un obstacle, c'est-à-dire d'un autre robot, et d'en déduire la violation des contraintes de précédence. Si le robot 2 détecte un obstacle, il se trouve alors en situation d'échec partiel et stoppe son déplacement. Suivant sa politique, il attendra ensuite plus ou moins longtemps avant de tenter à nouveau d'entrer sur le site.

La figure 11.4 présente le déroulement de la mission au niveau du site D (matérialisé par une croix noire sur le sol). Le robot 2 (dans la partie supérieure de l'image) arrive le premier à proximité du site. Anticipant le comportement du robot 1 et par conséquent le fait que celui-ci n'ait pas encore exploré le site D, le robot 2 attend pour pénétrer sur le site. Nous observons alors le robot 1 entrer sur le site D, y réaliser sa tâche (image 2) et en repartir (images 3 et 4). Rappelons que le robot 2 n'observe pas les actions entreprises par le robot 1, il ne peut constater la présence de ce dernier sur le site que s'il tente d'y entrer. Le robot 2 essaye donc ensuite d'entrer sur le site (image 5). Si les contraintes de précédence sont respectées, il achève son déplacement (image 6). En cas d'échec partiel, le robot 2 revient à sa position précédente et attend jusqu'à ce que sa politique lui dicte d'essayer à nouveau d'entrer sur le site. Une configuration similaire est observable pour les croisements des sites F et H.

La détection des échecs partiels est réalisée par observation de l'environnement via les capteurs infrarouges. Elle est, dans notre cas, cependant limitée par les effecteurs et les capteurs des robots utilisés. Si le robot 2 arrive en effet très en avance à proximité du site D et tente alors d'y entrer avant l'autre robot, il peut ne détecter aucun obstacle et conclut au respect des contraintes de précédence alors que celles-ci sont violées. L'utilisation de robots plus sophistiqués permet de résoudre de tels problèmes de perception en augmentant les capacités de communication indirecte des robots. Le robot 1 peut par exemple laisser une trace de son passage afin de signaler au robot 2 qu'il a déjà exploré le site. Lorsque le robot 2 tente d'entrer sur le site, la présence de cette marque lui permet alors de conclure sur le respect des contraintes de précédence.

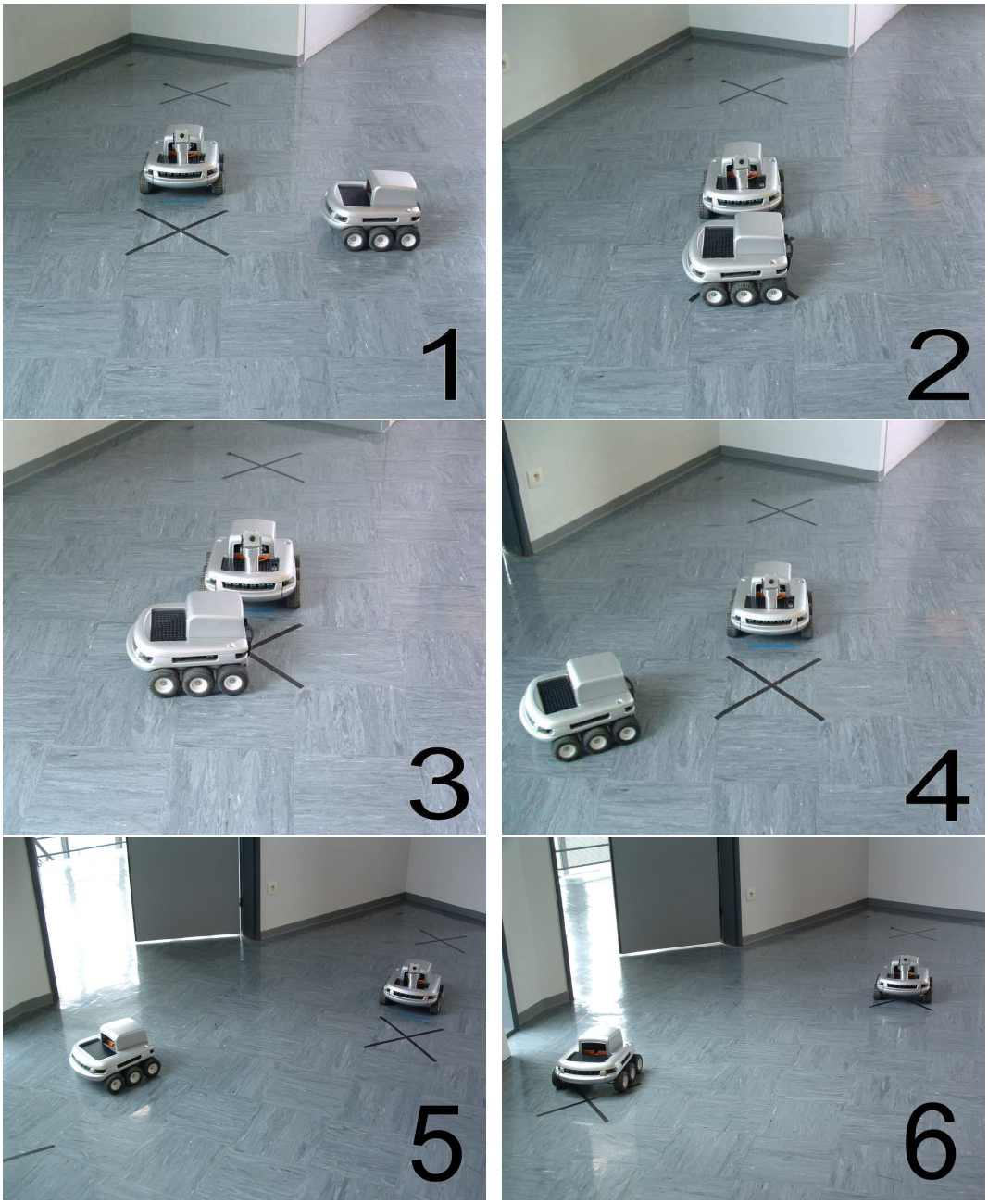


FIG. 11.4 – Exécution de la mission (croisement sur le site D)

11.2.3 Bilan des expérimentations sur les robots Koala

Les expérimentations réalisées sur le simulateur ont permis de vérifier que l'exécution des politiques conduisait effectivement au respect des différents types de contraintes. Ces mêmes politiques étant utilisées par les robots physiques, le respect des contraintes est par conséquent également garanti lors de l'exécution des tâches par les robots Koala.

En revanche, contrairement aux experimentations menées sur le simulateur dans des conditions idéales, l'exécution des missions par les robots Koala peut être interrompue par d'éventuels événements imprévus. L'application des politiques calculées hors-ligne suppose en effet, que les durées d'exécution des tâches et leurs consommations de ressources soient fidèles aux données du problème. Si un événement imprévu survient et entraîne une sur-consommation des ressources ou une durée d'exécution plus longue que celles envisagées, nous sortons du plan d'exécution calculé par notre approche et le robot se retrouve dans un état non envisagé. Il lui est alors impossible de continuer à exécuter sa politique et le robot se trouve dans une situation équivalente à un échec total. La prise en compte de l'incertitude permet toutefois de limiter ces cas d'échec par rapport aux approches ne traitant pas l'aspect stochastique des actions. Ce problème, inhérent à toute approche de planification hors-ligne, peut alors être résolu par une re-planification de l'exécution des tâches restantes en fonction de la nouvelle situation de l'agent.

Conclusion

Nous avons, dans ce chapitre, démontré l'applicabilité de notre approche pour la planification et l'exécution de missions multi-robots. A partir d'un scénario inspiré de la robotique exploratoire, nous avons présenté la démarche à suivre afin de procéder, en utilisant notre approche, à la planification et à l'exécution des tâches.

La mission considérée a tout d'abord été représentée sous forme d'un graphe orienté acyclique. Les algorithmes que nous avons développés ont ensuite permis, à partir de ce graphe, de procéder à la modélisation du problème sous forme d'OC-DEC-MDP puis, de calculer les politiques des agents. Nous nous sommes alors intéressée à l'exécution de ces politiques. L'application des politiques correspondant au parcours d'un automate à états finis, les capacités de calcul et la quantité de mémoire nécessaires par chaque robot restent limitées. Nous répondons ainsi aux exigences d'efficacité et de légèreté du processus délibératif.

L'exécution des missions en simulation nous a permis de vérifier l'applicabilité des politiques dans des conditions idéales d'exécution. L'utilisation des robots Koala nous a ensuite confrontée aux problèmes d'imprécision des capteurs et des effecteurs. Ces expérimentations ont toutefois permis de démontrer les possibilités d'exploitation des politiques calculées par des robots réels et ont prouvé qu'elles permettaient aux robots de mener à bien leur mission tout en respectant les contraintes sur l'exécution des tâches.

Conclusion de la partie IV

Cette quatrième partie a porté sur la validation de notre approche. Nous avons ainsi cherché à démontrer que les solutions proposées dans cette thèse permettent de répondre à la problématique initialement formulée.

L'étude des approches existantes réalisée au chapitre 3 a tout d'abord mis en évidence les limitations concernant la taille des problèmes traités. Nous avons ainsi commencé par tester l'efficacité des algorithmes que nous avons proposés. Il a été montré que notre approche permet de résoudre des problèmes composés de plusieurs centaines de tâches et dizaines d'agents. Dans un deuxième temps, nous nous sommes intéressée à la qualité des solutions retournées par nos algorithmes. Nous avons alors mis en évidence des dépendances entre la qualité de la solution, les méthodes d'estimation du coût occasionné et le temps de résolution. Nous avons toutefois pu constater que quelle que soit l'estimation du coût occasionné, les politiques calculées conduisent à de bonnes performances de la part des agents.

La seconde phase de validation a porté sur l'applicabilité de notre travail à des scénarios de la robotique collective. L'étude menée au chapitre 3 avait également mis en évidence les difficultés liées à la résolution de problèmes réels par les approches existantes. Nous avons donc, dans un deuxième chapitre, cherché à démontrer la capacité de notre approche à résoudre de tels problèmes. A partir d'un scénario représentatif de notre cadre applicatif, nous avons mis en évidence la possibilité de gérer les différents types de contraintes inventoriés et les différentes caractéristiques des problèmes envisagés (incertitude sur les durées d'exécution et les consommations de ressources, impossibilité de communiquer, etc.). Nous nous sommes ensuite intéressée à l'exécution des politiques sur des robots physiques et avons démontré l'applicabilité de nos solutions lors de la réalisation des tâches par de telles entités.

Toutes les données du scénario envisagé au précédent chapitre ont pu être formalisées par notre approche. Nous avons ainsi été en mesure de calculer des politiques respectant les différentes caractéristiques du problème. L'ensemble des contraintes que nous avons considérées dans cette thèse n'est bien entendu pas exhaustif. Néanmoins, des extensions de notre travail sont envisageables afin de permettre la formalisation d'une plus grande diversité de problèmes et ainsi offrir différentes perspectives à cette thèse.

Conclusion et Perspectives

Le travail présenté dans cette thèse a eu pour premier objectif de contribuer à atténuer l'écart actuellement existant entre les besoins suscités par le développement d'applications multi-agents (les colonies de robots autonomes par exemple) et les solutions apportées par les recherches en théorie de la décision. Nous nous sommes plus particulièrement intéressée aux approches basées sur les Processus Décisionnels de Markov Décentralisés et sommes partie du constat que très peu de ces travaux pouvaient être utilisés dans des applications réelles.

Conclusion Générale

Une première partie bibliographique nous a permis de poser le problème de la prise de décision sous incertitude dans les systèmes multi-agents coopératifs. Nous avons ainsi présenté les difficultés liées à la décision décentralisée dans de tels systèmes. Nous avons alors insisté sur la nécessaire coordination des décisions malgré le manque d'observabilité et sur l'importance des interactions entre agents. Nous avons ensuite introduit un formalisme mathématique dédié à la modélisation de tels problèmes : les Processus Décisionnels de Markov Décentralisés. L'étude des approches dédiées à leur résolution nous a permis de mettre en évidence un certain nombre de faiblesses[54] à la fois dans la modélisation du problème et dans les méthodes de résolution proposées :

- Les DEC-MDPs ne permettent qu'une modélisation restreinte des données temporelles, chaque tâche ne durant qu'une unité de temps.
- Les DEC-MDPs ne prennent pas en compte d'éventuelles contraintes sur l'exécution des tâches.
- Les complexités des algorithmes de résolution existants sont telles qu'elles limitent la taille des problèmes pouvant être envisagés.

Ainsi les DEC-MDPs proposent un formalisme adapté à la résolution de problèmes académiques mais nous confrontent à différentes difficultés lorsqu'il s'agit de résoudre des problèmes

[54] Ces faiblesses sont en rapport avec l'utilisation des DEC-MDPs pour la planification dans des applications réelles, en particulier dans les colonies de robots autonomes.

de décision multi-agents tels que ceux rencontrés dans les applications réelles.

Partant de ce constat, nos travaux ont été motivés par deux principaux objectifs :

- permettre une modélisation plus riche du temps et des actions dans les Processus Décisionnels de Markov Décentralisés,
- augmenter l'efficacité de la résolution des problèmes envisagés.

La suite de cette thèse a alors été organisée autour de ces deux axes. Nous avons tout d'abord procédé à une analyse des problèmes que nous souhaitions résoudre afin d'en proposer une modélisation adaptée. Nous nous sommes ensuite tournée vers la résolution de ces problèmes. Enfin, nous avons procédé à la validation de notre approche dans le but de vérifier que les solutions proposées répondaient aux objectifs précédemment cités.

Enrichissement de la modélisation du temps et des actions

Après avoir analysé les problèmes de décision multi-agents que nous souhaitions envisager, nous nous sommes concentrée sur leur modélisation en vue d'une résolution efficace. Nous avons alors enrichi la modélisation du temps proposée par les DEC-MDPs et avons permis la prise en compte de contraintes temporelles, de précédence et de ressources.

Afin de gérer des scénarios constitués de plusieurs centaines de tâches et dizaines d'agents, nous avons procédé à une décomposition du problème initial en un ensemble de MDPs représentant chacun le problème décisionnel d'un agent. Nous avons ainsi décrit une modélisation du problème adaptée à nos besoins et introduit un nouveau formalisme : l'OC-DEC-MDP.

Dans le but de définir de manière complète chaque MDP constituant un OC-DEC-MDP, il a été nécessaire de décomposer la fonction de transition du système en un ensemble de fonctions propres à chaque agent. Les dépendances entre agents étant modélisées *via* cette fonction de transition, une telle décomposition n'est *a priori* pas triviale. Elle a toutefois été rendue possible en supposant la politique de chaque agent connue et fixée. Nous avons ainsi pu déduire les probabilités de transition de chacun et déduire la fonction de transition de chaque MDP.

Résolution polynomiale du problème de décision multi-agent

Une fois la modélisation du problème achevée, nous nous sommes intéressée à sa résolution. Étant donnée la complexité de la résolution optimale d'un tel problème, nous nous sommes orientée vers la recherche d'une approximation de la politique optimale, c'est-à-dire une solution répondant au critère de rationalité limitée. Nous avons alors cherché à calculer une solution par amélioration de l'ensemble des politiques initialement fixées et utilisées lors de la construction de l'OC-DEC-MDP.

La décomposition du problème en un ensemble de MDPs nous a par ailleurs confrontée à la question de la coordination entre les agents, cette dernière étant nécessaire afin d'obtenir des

politiques de qualité et coordonnées. Nous avons alors fait appel à une notion provenant des sciences économiques, le coût occasionné, permettant de mesurer l'impact d'une décision sur les autres agents. Chaque agent a ainsi été en mesure de décider quelle action exécuter à partir d'un quelconque état, de façon à maximiser le gain espéré collectif.

Un premier algorithme de résolution polynomial a alors été développé. Nous avons pour ce faire cherché à limiter les mises à jour du modèle tout en augmentant le plus possible la qualité des politiques. L'itération du processus d'amélioration ensuite mis en place a permis d'accroître les performances de notre approche en garantissant l'obtention d'une situation d'équilibre. Nous avons également identifié différentes propriétés assurant le calcul d'une solution optimale.

Bilan

L'étude de complexité de nos algorithmes, les garanties portant sur la qualité des solutions ainsi que les expérimentations menées lors de la validation de notre approche, nous ont permis de démontrer l'adéquation entre nos objectifs initialement fixés et les solutions que nous avons proposées.

Nous avons en effet développé des algorithmes de résolution efficaces tenant compte des propriétés du problème et menant à l'obtention d'une situation d'équilibre[55]. Cette dernière nous semble pouvoir correspondre à un équilibre de Nash. Nous nous intéressons donc actuellement aux propriétés de cet équilibre.

A la différence des travaux portant actuellement sur les DEC-MDPs, notre approche présente l'avantage de pouvoir traiter des problèmes de taille importante tels que ceux rencontrés dans les applications réelles. La prise en compte des contraintes sur l'exécution des tâches a contribué au traitement de grands problèmes. Ces contraintes ont en effet pour première conséquence de complexifier la décision (respect des contraintes, interactions entre les agents, etc.). Elles contribuent cependant également à limiter les plans d'exécution possibles et ainsi restreignent les espaces d'états et d'actions.

Les travaux que nous avons présentés contribuent par ailleurs à augmenter l'applicabilité des Processus Décisionnels de Markov. Nous avons en effet permis la prise en compte des différentes contraintes devant être respectées dans les problèmes réels. Toutes les caractéristiques des applications réelles n'ont toutefois pas été considérées et l'enrichissement des données pouvant être formalisées offre de nombreuses perspectives à notre travail.

[55] Pour la version itérative.

Perspectives de recherche

Une première partie des perspectives de ce travail a pour objectif d'augmenter l'espace des problèmes pouvant être résolus. Pour ce faire, nous envisageons plus particulièrement de nous intéresser à :

- **l'enrichissement des contraintes :** nous souhaitons tout d'abord augmenter la diversité des contraintes pouvant être prises en compte par notre approche et enrichir la représentation des contraintes déjà considérées. Il est par exemple envisageable d'étendre la formalisation des contraintes temporelles portant sur les dates de début et de fin des tâches, à un ensemble d'intervalles convexes. La diversification des contraintes peut par ailleurs être guidée par l'ensemble des relations définies dans le formalisme TAEMS [Decker et Lesser, 1993b]. Nous pouvons ainsi considérer des dépendances de qualité entre les tâches ou bien relâcher les contraintes de précédence pour définir des relations du type « la réalisation de la tâche t_i facilite l'exécution de la tâche t_j ». Des relations imposant la simultanéité de plusieurs tâches peuvent également être envisagées.
- **l'enrichissement de la topologie du graphe :** l'ajout de nœuds « ou » dans le graphe ainsi que la possibilité de « passer » des tâches permettraient également d'augmenter la généricité des problèmes et de diversifier les possibilités d'exécution d'une mission. Il serait alors possible de choisir entre l'exécution d'une tâche t_i ou d'une tâche t_j et d'ignorer certaines tâches afin d'en exécuter d'autres plus importantes. La substitution des tâches par des unités de raisonnement progressif (PRU) [Mouaddib et Zilberstein, 1998, Zilberstein et Mouaddib, 1999, Cardon *et al.*, 2001] offre également une alternative intéressante à l'enrichissement de la topologie du graphe. Chaque PRU devrait alors être considérée mais les tâches la constituant pourraient être exécutées suivant différents plans.

La mise en œuvre de ces perspectives nécessitera de réviser notre modèle. En effet, la description des états, des actions et la définition de la fonction de transition devront très probablement être adaptées afin de modéliser au mieux ces nouveaux problèmes.

L'étude des problèmes de coordination entre agents a par ailleurs démontré l'importance des échecs partiels et leur impact sur les performances du système. Dans les cas litigieux où les échecs partiels sont préjudiciables et où retarder l'exécution de la prochaine tâche n'est pas souhaitable, nous pouvons envisager l'ajout de communication durant l'exécution[56]. En effet, si le coût de la communication est inférieur au coût d'un échec partiel, il peut être préférable de communiquer afin de connaître l'état d'avancement des prédécesseurs. Nous envisageons d'étudier plus en détails les perspectives qu'offrirait une telle communication en ce qui concerne l'amélioration des

[56] Ceci n'est toutefois réalisable que dans certaines applications où la communication est physiquement possible.

performances des agents. Cette piste de recherche nous conduira à plus long terme à étudier la question de la communication lors de la prise de décision dans les systèmes multi-agents. Un tel échange d'informations permet en effet d'améliorer les décisions [Nair *et al.*, 2004] mais possède toutefois un coût. Comme le montrent [Goldman et Zilberstein, 2004, Becker *et al.*, 2005], il est alors nécessaire de déterminer si les connaissances acquises en communiquant apportent un bénéfice supérieur au coût de la communication.

Enfin, le travail présenté dans cette thèse s'est essentiellement appuyé sur des applications liées à la robotique mobile. Nous pensons toutefois que notre travail n'est pas restreint à ce cadre. Nous souhaitons donc également étendre l'applicabilité de notre approche à des projets plus divers.

Annexes

Annexe A

Preuves des théorèmes énoncés

A.1 Chapitre 6

A.1.1 Corollaire 1

$P_{suc} + P_{PCV} + P_{LR} + P_{TL} + P_{DM} = 1$ *donc le système de transition est complet.*

Preuve :

1) Supposons que $st < UB$:

$$P_{suc} + P_{PCV} + P_{MR} + P_{TT} + P_{DM} =$$

$$P^{r}_{enough}(Pred(t_{i+1})).\prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k)|t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}).\sum_{\Delta_r^{i+1}|r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1}|st+\delta_c^{i+1} \leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1}) \tag{A.1}$$

$$+ \sum_{\Delta r'|r \geq \Delta r'} P_r(\Delta r') \cdot P_{not\_end}(st) \tag{A.2}$$

$$+ P^{r}_{enough}(Pred(t_{i+1})).\prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k)|t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}) \sum_{\Delta_r^{i+1}|r < \Delta_r^{i+1}} P_r(\Delta_r^{i+1}) \tag{A.3}$$

$$+ P_{not\_end}(st).\sum_{\Delta'_r|r < \Delta'_r} P_r(\Delta'_r) \tag{A.4}$$

$$+ P^{r}_{enough}(Pred(t_{i+1})).\prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k)|t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}).\sum_{\Delta r|r \geq \Delta_r^{i+1}} P_r(\Delta_r^{i+1}).\sum_{\delta_c^{i+1}|st+\delta_c^{i+1} > LET} P_c(\delta_c^{i+1}) \tag{A.5}$$

Des simplifications peuvent être réalisées en regroupant les termes de cette somme. En par-

ticulier :

$$(A.2) + (A.4) = P_{not\_end}(st) \tag{A.6}$$

$$(A.1) + (A.5) = P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}). \sum_{\Delta^{i+1}_r | r \geq \Delta^{i+1}_r} P_r(\Delta^{i+1}_r). \tag{A.7}$$

$$(A.7) + (A.3) = P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}) \tag{A.8}$$

Par ailleurs,

$$P_{not\_end}(st) = P^r_{enough}(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1}) - t_i} \sum_{end \in ET(t_k) | et \leq UB} P^{t_k}_{et}(end|et(I')_{t_i}) \tag{A.9}$$

$$- P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}) \tag{A.10}$$

$$+ 1 - P^r_{enough}(Pred(t_{i+1})) \cdot \prod_{t_k \in P_{red}(t_{i+1})} \sum_{end \in ET(t_k) | et \leq UB} P^{t_k}_{et}(end|et(I')_{t_i}) \tag{A.11}$$

Enfin,

$$(A.8) + (A.10) = 0 \tag{A.12}$$

et

$$(A.9) + (A.11) = 1 \tag{A.13}$$

D'où :

$$P_{suc} + P_{PCV} + P_{MR} + P_{TT} + P_{DM} = 1 \tag{A.14}$$

2) Supposons que $st \geq UB$:

$$P_{suc} + P_{PCV} + P_{MR} + P_{TT} + P_{DM} =$$

$$P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}). \sum_{\Delta^{i+1}_r | r \geq \Delta^{i+1}_r} \sum_{\delta^{i+1}_c | st + \delta^{i+1}_c \leq LET} P_r(\Delta^{i+1}_r).P_c(\delta^{i+1}_c) \tag{A.15}$$

$$+ P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}) \sum_{\Delta^{i+1}_r | r < \Delta^{i+1}_r} P_r(\Delta^{i+1}_r) \tag{A.16}$$

$$+ P_{not\_end}(st) \tag{A.17}$$

$$+ P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k) | t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}). \sum_{\Delta^{i+1}_r | r \geq \Delta^{i+1}_r} P_r(\Delta^{i+1}_r). \sum_{\delta^{i+1}_c | st + \delta^{i+1}_c > LET} P_c(\delta^{i+1}_c) \tag{A.18}$$

Comme précédemment :

$$(A.15) + (A.18) = P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k)|t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}). \sum_{\Delta^{i+1}_r | r \geq \Delta^{i+1}_r} P_r(\Delta^{i+1}_r). \tag{A.19}$$

$$(A.19) + (A.16) = P^r_{enough}(Pred(t_{i+1})). \prod_{t_k \in P_{red}(t_{i+1})} \sum_{t \in ET(t_k)|t \leq st} P^{t_k}_{et}(t|et(I')_{t_i}) \tag{A.20}$$

$$(A.20) + (A.10) = 0 \tag{A.21}$$

et

$$(A.9) + (A.11) = 1 \tag{A.22}$$

D'où :

$$P_{suc} + P_{PCV} + P_{MR} + P_{TT} + P_{DM} = 1 \tag{A.23}$$

A.2 Chapitre 8

A.2.1 Preuve du corollaire 9

L'action a_i maximisant l'équation 7.4 maximise également l'utilité espérée jointe des agents.

Preuve :

Rappelons tout d'abord l'équation permettant de déterminer la politique EOC :

$$\pi_i(s_i) = arg\_max_{E(t_{i+1},st),st \geq t}\Big(\overbrace{V'}^{\text{Utilité espérée}} - \overbrace{OC(t_{i+1}, st)}^{\text{Coût occasionné}} \Big)$$

Lorsque le coût occasionné est correctement estimé, il reflète précisément la perte en utilité espérée des autres agents. En effet, nous avons :

$$\begin{aligned} OC(t_{i+1}, st) = & \overbrace{P^r_{enough}(Pred(t_{i+1})). \prod_{a \in P_{red}(t_{i+1}) - t_i} \sum_{s \leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}} \\ & \overbrace{. \sum_{\Delta r | r \geq \Delta^{i+1}_r} \sum_{\delta^{i+1}_c | st + \delta^{i+1}_c \leq LET} P_r(\Delta^{i+1}_r).P_c(\delta^{i+1}_c)}^{\text{Probabilité de succès}} . \sum_{\mathcal{A}g_j \in \text{other\_agents}} EOC_{\mathcal{A}g_j, t_{i+1}}(et_{i+1}) \\ & + \overbrace{(P_{MR} + P_{TT} + P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j \in \text{other\_agents}} EOC_{\mathcal{A}g_j, t_{i+1}}(fail) \\ & + \overbrace{P_{PPC}}^{\text{Probabilité d'échouer partiellement}} \cdot OC(t_{i+1}, next\_start) \end{aligned}$$

où

$$EOC_{\mathcal{A}g_j,t_{i+1}}(fail) = OC_{t_j}(fail) = \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^0_{t_j,r_{t_j}} - V([failure_{t_j},*,*])$$

Le coût occasionné peut donc être reformulé de la manière suivante :

$$\begin{aligned} OC(t_{i+1},st) = & \overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a\in P_{red}(t_{i+1})-t_i}\sum_{s\leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}} \\ & \overbrace{.\sum_{\Delta r|r\geq\Delta^{i+1}_r}\sum_{\delta^{i+1}_c|st+\delta^{i+1}_c\leq LET} P_r(\Delta^{i+1}_r).P_c(\delta^{i+1}_c)}^{\text{Probabilité de succès}} \cdot \sum_{\mathcal{A}g_j\in \text{other\_agents}} EOC_{\mathcal{A}g_j,t_{i+1}}(et_{i+1}) \\ & + \overbrace{(P_{MR}+P_{TT}+P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j\in \text{other\_agents}} (\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^0_{t_j,r_{t_j}} - V([failure_{t_j},*,*])) \\ & + P_{PPC-fail} \sum_{\mathcal{A}g_j\in \text{other\_agents}} (\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^0_{t_j,r_{t_j}} - V([failure_{t_j},*,*])) \\ & + \sum_{et'(t_{i+1})>et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1})) \sum_{\mathcal{A}g_j\in \text{other\_agents}} EOC_{\mathcal{A}g_j,t_{i+1}}(et'(t_{i+1})*) \end{aligned}$$

où $P_{PPC-fail}$ désigne la probabilité que l'exécution de la tâche t_{i+1} échoue après 1 échec partiel ou plus et $P_{PPC-suc}(et'(t_{i+1}))$ désigne la probabilité que l'exécution de la tâche t_{i+1} réussisse après 1 échec partiel ou plus et finisse à $et'(t_{i+1})$. Nous avons ainsi :

$$P_{PPC} = \sum_{et'(t_{i+1})>et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1})) + P_{PPC-fail}$$

Par ailleurs,

$$EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}}) = max_{t_k\in Suc(t_{i+1})} \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) OC_j(et_{t_{i+1}} - LB_k, r_{t_j})$$

Afin de limiter la longueur des formules, nous supposerons que $t_j = t_k$. Dans le cas contraire, il est possible de revenir à une démonstration équivalente en appliquant récursivement les équations suivantes :

$$EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}}) = max_{t_k\in Suc(t_j)} EOC_{t_k}(et_{t_{i+1}} - LB_k, t_j)$$

$$EOC_{t_k}(et_{t_i} - LB_k, t_j) = \sum_{r_{t_k}} P^{t_k}_{ra}(r_{t_k}) \cdot \sum_{\delta_c} P^{t_k}_c(\delta_c) max_{t_l\in Suc(t_k)} EOC_{t_l}(st^* + \delta_c - LB_l, t_j)$$

Nous en déduisons que :

$$\begin{aligned}
EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}}) &= max_{t_j \in Suc(t_{i+1})} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) OC_j(et_{t_i} - LB_k, r_{t_j}) \\
&= max_{t_j \in Suc(t_{i+1})} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j})(V_{t_j}^{0,r_{t_j}} - V_{t_j}^{\Delta t, r_{t_j}}) \\
&= max_{t_j \in Suc(t_{i+1})} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{0,r_{t_j}} - \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}} \\
&= \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) \cdot V_{t_j}^{0,r_{t_j}} - min_{t_j \in Suc(t_{i+1})} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}}
\end{aligned}$$

Lorsque plusieurs tâches $t_j \in Suc(t_{i+1})$ provoquent différents retards sur une même tâche, l'opérateur « min » (ou « max » suivant le signe précédant l'opérateur) garantit la prise en compte du plus grand retard. En effet, lorsqu'une tâche t_l retarde une tâche t_j de Δt et qu'une autre tâche $t_{l'}$ retarde t_j de $\Delta t'$, le retard provoqué sur t_j correspond au maximum entre Δt et $\Delta t'$. Pour des raisons de concision des formules, nous supposerons par la suite que le retard maximum est effectivement bien pris en compte et omettrons l'opérateur « min ».

Nous obtenons alors :

$$EOC_{\mathcal{A}g_j,t_{i+1}}(et_{t_{i+1}}) = \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{0,r_{t_j}} - \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}}$$

où $\Delta t = et_{t_{i+1}} - LB_j$

$$OC(t_{i+1}, st) = \overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a \in P_{red}(t_{i+1})-t_i} \sum_{s \leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}}$$

$$\overbrace{. \sum_{\Delta r | r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1} | st+\delta_c^{i+1} \leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}} \cdot$$

$$\sum_{\mathcal{A}g_j \in \text{other\_agents}} (\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^{0,r_{t_j}}_{t_j} - \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^{\Delta t, r_{t_j}}_{t_j})$$

$$+ \overbrace{(P_{MR} + P_{TT} + P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^0_{t_j, r_{t_j}}$$

$$- \overbrace{(P_{MR} + P_{TT} + P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j \in \text{other\_agents}} V([failure_{t_j}, *, *])$$

$$+ P_{PPC-fail} \sum_{\mathcal{A}g_j \in \text{other\_agents}} (V^0_{t_j} - V([failure_{t_j}, *, *]))$$

$$+ \sum_{et'(t_{i+1}) > et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1})) \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^{0,r_{t_j}}_{t_j}$$

$$- \sum_{et'(t_{i+1}) > et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1})) \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V^{\Delta t, r_{t_j}}_{t_j}$$

Soit :

$$
\begin{aligned}
OC(t_{i+1}, st) = & \overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a\in P_{red}(t_{i+1})-t_i}\sum_{s\leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}} \\
& \overbrace{.\sum_{\Delta r|r\geq\Delta_r^{i+1}}\sum_{\delta_c^{i+1}|st+\delta_c^{i+1}\leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}}.\sum_{\mathcal{A}g_j\in \text{other\_agents}}\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})V^{0,r_{t_j}}_{t_j} \\
& + \overbrace{(P_{MR}+P_{TT}+P_{DM})}^{\text{Probabilité d'échouer totalement}}\sum_{\mathcal{A}g_j\in \text{other\_agents}}\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})V^{0}_{t_j,r_{t_j}} \\
& + \sum_{et'(t_{i+1})>et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1}))\sum_{\mathcal{A}g_j\in \text{other\_agents}}\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})V^{0,r_{t_j}}_{t_j} \\
& + P_{PPC-fail}\sum_{\mathcal{A}g_j\in \text{other\_agents}}\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})V^{0}_{t_j,r_{t_j}} \\
& - \overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a\in P_{red}(t_{i+1})-t_i}\sum_{s\leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}} \\
& \overbrace{.\sum_{\Delta r|r\geq\Delta_r^{i+1}}\sum_{\delta_c^{i+1}|st+\delta_c^{i+1}\leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}}.\sum_{\mathcal{A}g_j\in \text{other\_agents}}\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})V^{\Delta t,r_{t_j}}_{t_j}) \\
& - \overbrace{(P_{MR}+P_{TT}+P_{DM})}^{\text{Probabilité d'échouer totalement}}\sum_{\mathcal{A}g_j\in \text{other\_agents}} V([failure_{t_j},*,*]) \\
& - P_{PPC-fail}\sum_{\mathcal{A}g_j\in \text{other\_agents}} V([failure_{t_j},*,*]) \\
& - \sum_{et'(t_{i+1})>et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1}))\sum_{\mathcal{A}g_j\in \text{other\_agents}}\sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j})V^{\Delta t,r_{t_j}}_{t_j}
\end{aligned}
$$

A partir du corollaire 1 prouvant la complétude du système de transition, nous en déduisons que :

$$
\overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a\in P_{red}(t_{i+1})-t_i}\sum_{s\leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i}).\sum_{\Delta r|r\geq\Delta_r^{i+1}}\sum_{\delta_c^{i+1}|st+\delta_c^{i+1}\leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}}
$$

$$
+P_{MR}+P_{TT}+P_{DM}+P_{PPC}=1
$$

D'où la simplification qui suit :

$$
\begin{aligned}
OC(t_{i+1}, st) = & \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{0, r_{t_j}} \\
& - \overbrace{P_{enough}^{r}(Pred(t_{i+1})).\prod_{a \in P_{red}(t_{i+1}) - t_i} \sum_{s \leq st} P_{\mathcal{ET}}^{a}(s | et(I')_{t_i})}^{\text{Probabilité de succès}} \\
& \overbrace{\cdot \sum_{\Delta r | r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1} | st + \delta_c^{i+1} \leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}} \cdot \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}}) \\
& - \overbrace{(P_{MR} + P_{TT} + P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j \in \text{other\_agents}} V([failure_{t_j}, *, *]) \\
& - P_{PPC-fail} \sum_{\mathcal{A}g_j \in \text{other\_agents}} V([failure_{t_j}, *, *]) \\
& - \sum_{et'(t_{i+1}) > et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1})) \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}}
\end{aligned}
$$

Quelle que soit la date de début st de t_{i+1}, la distribution de probabilité sur les taux de ressources r_{t_j} reste inchangée. $\sum_{r_{t_j}} P_{ra}^{t_j}(r_{t_j}) V_{t_j}^{0, r_{t_j}}$ est donc constant quelle que soit la date de début de t_{i+1}.

Par ailleurs, si le coût occasionné est estimé de manière exacte et précise, il traduit fidèlement l'utilité espérée des agents. Rappelons que :

$$
V_{t_j}^{\Delta t, r_{t_j}} = \sum_{\Delta_r^j} \sum_{\delta_c^j} P_r(\Delta_r^j) P_c(\delta_c^j).V[t_j, [st(LB_{t_j} + \Delta t), st(LB_{t_j} + \Delta t) + \delta_c^j], r_{t_j} - \Delta_r^j - \Delta r' \cdot nbEP]
$$

Le coût occasionné peut donc se ré-écrire comme la différence entre un terme constant (C) et l'utilité espérée des autres agents. Ce que nous pouvons représenter par l'équation suivante :

$$
OC(t_{i+1}, st) = C - \sum_{\mathcal{A}g_j \in \text{other\_agents}} V'_{\mathcal{A}g_j}
$$

où $V'_{\mathcal{A}g_j}$ désigne l'utilité espérée de l'agent $\mathcal{A}g_j$.

En effet, puisque le coût occasionné est estimé de manière exacte, nous avons :

$$\sum_{\mathcal{A}g_j \in \text{other\_agents}} V'_{\mathcal{A}g_j} = \overbrace{P^r_{enough}(Pred(t_{i+1})).\prod_{a \in P_{red}(t_{i+1})-t_i} \sum_{s \leq st} P^a_{\mathcal{ET}}(s|et(I')_{t_i})}^{\text{Probabilité de succès}}$$
$$\overbrace{\cdot \sum_{\Delta r | r \geq \Delta_r^{i+1}} \sum_{\delta_c^{i+1} | st+\delta_c^{i+1} \leq LET} P_r(\Delta_r^{i+1}).P_c(\delta_c^{i+1})}^{\text{Probabilité de succès}} \cdot \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}})$$
$$+ \overbrace{(P_{MR} + P_{TT} + P_{DM})}^{\text{Probabilité d'échouer totalement}} \sum_{\mathcal{A}g_j \in \text{other\_agents}} V([failure_{t_j}, *, *])$$
$$+ P_{PPC-fail} \sum_{\mathcal{A}g_j \in \text{other\_agents}} V([failure_{t_j}, *, *])$$
$$+ \sum_{et'(t_{i+1}) > et(t_{i+1})} P_{PPC-suc}(et'(t_{i+1})) \sum_{\mathcal{A}g_j \in \text{other\_agents}} \sum_{r_{t_j}} P^{t_j}_{ra}(r_{t_j}) V_{t_j}^{\Delta t, r_{t_j}}$$

L'équation calculant la politique d'une agent revient donc à maximiser l'ensemble des termes suivants :

$$\overbrace{V'}^{\text{Utilité espérée}} - \overbrace{C}^{\text{Constante}} + \overbrace{\sum_{\mathcal{A}g_j \in \text{other\_agents}} V'_{\mathcal{A}g_j}}^{\text{Utilité espérée des autres agents}}$$

L'action sélectionnée maximise donc l'utilité espérée collective. Lorsqu'un agent modifie sa politique nous pouvons par conséquent garantir que la nouvelle action considérée augmente les performances du système. □

A.2.2 Preuve du théorème 2

Si les ressources des agents sont illimitées alors, la politique EST est une politique optimale jointe.

Preuve : Si les ressources sont illimitées, les décisions ne sont pas influencées par la quantité de ressources disponibles. Les états de succès peuvent donc être réduits à des couples $[t_i, I]$ et les états d'échec partiel à des triplets $[t_i, et(I'), I]$. Les probabilités de transition ne sont par ailleurs pas influencées par les probabilités sur les taux de ressources et les consommations de ressources.

Rappelons que les agents cherchent à maximiser la somme de leurs utilités espérées. Soit π_i la politique individuelle de l'agent $\mathcal{A}g_i$. La politique optimale π^* des agents est définie par l'équation suivante :

$$\pi^* = arg\_max_{\pi = \langle \pi_1, \pi_2, \cdots, \pi_n \rangle} \sum_i V^{\pi_i}$$

où V^{π_i} correspond à l'utilité espérée de l'agent $\mathcal{A}g_i$ lorsqu'il suit la politique π_i.

Nous montrerons, tout d'abord, que si les ressources sont illimitées, la politique EST maximise l'utilité espérée de chaque agent. Ainsi, la politique EST est une politique individuellement

optimale. Nous raisonnerons par récurrence afin de démontrer que si un agent commence l'exécution de ses tâches au plus tôt alors, il maximise son utilité espérée. Nous montrerons que cette hypothèse est vraie pour la dernière tâche d'un agent (initialisation du raisonnement par récurrence). Nous supposerons alors l'hypothèse vérifiée pour la $n^{\text{ème}}$ tâche d'un agent $\mathcal{A}g_j$ et montrerons qu'elle s'avère alors également vraie pour la $n-1^{\text{ème}}$ tâche exécutée $\mathcal{A}g_j$.

Soit $\mathcal{A}g_j$ un agent devant exécuter une tâche t_j. Soient st la première date de début possible pour l'exécution de t_j (date de début dictée par la politique EST) et st' une date de début possible de t_j telle que $st' > st$. V_{st} désigne l'utilité espérée de l'agent $\mathcal{A}g_j$ lorsqu'il suit la politique EST et qu'il commence par conséquent l'exécution de la tâche t_j à st. Si l'agent échoue partiellement, il tentera à nouveau d'exécuter la tâche le plus tôt possible. Si $st+1$ est une date de début possible pour t_j, l'agent tentera donc à nouveau d'exécuter la tâche à $st+1$. $V_{st'}$ désignera l'utilité espérée de l'agent lorsque celui-ci commence l'exécution de t_j à st' au lieu de suivre la politique EST.

Nous souhaitons alors montrer que $V_{st} \geq V_{st'}\ \forall st'$.

1) Si t_j est la dernière tâche de l'agent $\mathcal{A}g_j$:

Lorsque l'agent $\mathcal{A}g_j$ commence l'exécution de la tâche t_j à st, il peut réussir son exécution, échouer partiellement ou totalement.

$$V_{st} = P^{st}_{suc}V^{st}_{suc} + P^{st}_{PCV}V^{st}_{PCV} + P^{st}_{fail}V^{st}_{fail}$$

où $P^{st}_{fail} = P^{st}_{LR} + P^{st}_{TL} + P^{st}_{DM}$

V^{st}_{suc} dépend des durées d'exécution de la tâche t_j. P^{st}_{suc} correspond à la probabilité que la tâche soit exécutée avec succès en commençant à st, P^{st}_{PCV} désigne la probabilité que l'agent échoue partiellement en commençant à st, et P^{st}_{fail} représente la probabilité que l'exécution échoue totalement en commençant à st.

Si les ressources sont illimitées, $P^{st}_{fail} = P^{st}_{TL} + P^{st}_{DM}$.

De plus, si $st < UB$, $P^{st}_{fail} = P^{st}_{TL}$.

V^{st}_{fail} est par ailleurs défini de la façon suivante : $V^{st}_{fail} = V([failure_{t_j}, *, *])$

t_j étant la dernière tâche de l'agent, nous avons $V^{st}_{suc} = \mathcal{R}_{t_j}\ \forall st$ et $V^{st}_{fail} = -\mathcal{R}_{t_j}\ \forall st$.

Lorsque l'agent échoue partiellement, il peut essayer à nouveau d'exécuter la tâche plus tard et ainsi réussir, échouer à nouveau partiellement ou échouer totalement.

V_{st} peut donc être décomposé de la façon suivante :

$$V_{st} = V^{st}_{SUC} + V^{st}_{FAIL}$$

où V^{st}_{SUC} regroupe l'ensemble des exécutions réussies de t_j et V^{st}_{FAIL} représente les exécutions conduisant à un échec total. De plus,

$$V^{st}_{FAIL} = P^{st}_{FAIL}V^{st}_{fail}$$

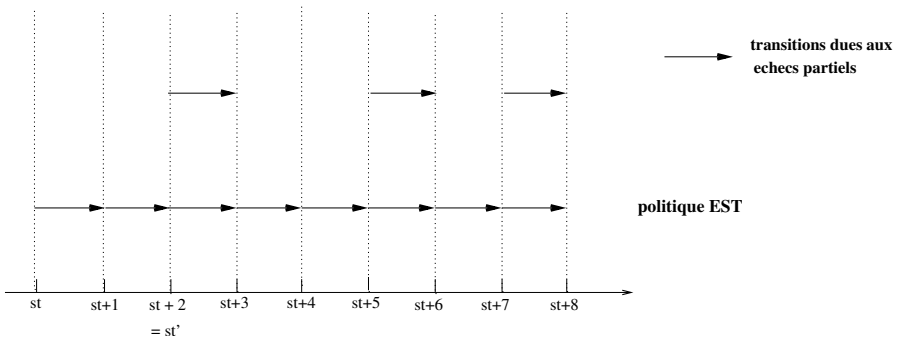


FIG. A.1 – Transitions dues aux échecs partiels

où P^{st}_{FAIL} désigne la probabilité que l'exécution de t_j échoue totalement.

La figure A.1 présente les différentes transitions dues aux échecs partiels. Afin de clarifier les notations, nous supposerons que tout instant t, $t \in [st, UB_j]$, correspond à une date de début possible pour t_j. Les équations qui suivent peuvent néanmoins être adaptées aux cas où il existe un instant t', $t' \in [st, UB_j]$, ne correspondant pas à une date de début de t_j. Si l'agent suit la politique EST, il commencera à exécuter la tâche à st. Si il échoue partiellement, il tentera à nouveau l'exécution à $st+1$ (la prochaine date de début la plus proche). Si l'agent échoue une nouvelle fois partiellement, il essayera à nouveau d'exécuter la tâche à $st+2$ et ainsi de suite. Après k échecs partiels, l'agent tentera l'exécution de la tâche à $st' = st + k$. Ainsi, V^{st}_{SUC} et P^{st}_{FAIL} peuvent être décomposés de la façon suivante :

$$\begin{aligned} V^{st}_{SUC} =& P^{st}_{suc} V^{st}_{suc} + \sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{t}_{suc} + (\prod_{t'<st'} P^{t'}_{PCV}) P^{st'}_{suc} V^{st'}_{suc} \\ &+ \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{t}_{suc} \end{aligned}$$

où t' et t correspondent à des dates de début possibles pour la tâche t_j

$$\begin{aligned} P^{st}_{FAIL} =& P^{st}_{DM} + \overbrace{\sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV})(P^{t}_{DM} - P^{t-1}_{DM})}^{A_1} + \overbrace{(\prod_{t'<st'} P^{t'}_{PCV})(P^{st'}_{DM} - P^{st'-1}_{DM})}^{B_1} \\ &+ \overbrace{\sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV})(P^{t}_{DM} - P^{t-1}_{DM})}^{C_1} + P^{UB}_{TL} \end{aligned}$$

$st' - 1$ correspond à la précédente date de début à laquelle l'agent a tenté d'exécuter la tâche. S'il existe un instant t', $t' \in [st, UB_j]$, tel que t' ne soit pas une date de début de t_j, la précédente date de début pourra être notée st'^{-} au lieu de $st' - 1$.

De façon similaire,

$$V_{st'} = P^{st'}_{suc} V^{st'}_{suc} + P^{st'}_{PCV} V^{st'}_{PCV} + P^{st'}_{fail} V^{st'}_{fail} = V^{st'}_{SUC} + V^{st'}_{FAIL}$$

Lorsque l'agent $\mathcal{A}g_j$ commence la tâche t_j à st', il peut réussir l'exécution, échouer partiellement ou totalement. En cas d'échec partiel, si l'agent ne suit pas la politique EST, il est susceptible de tenter à nouveau l'exécution de la tâche à $st' + k$ où $st' + k > st' + 1$. L'agent peut alors échouer partiellement à $st' + k$, et tenter d'exécuter à nouveau la tâche à $st' + k + j$. La partie supérieure de la figure A.1 représente les transitions dues aux échecs partiels lorsque l'agent suit une telle politique. Dans cet exemple, l'agent commence à exécuter la tâche à st', il échoue partiellement et tente à nouveau l'exécution de la tâche à $st+5$. S'il échoue partiellement à $st+5$, il essaye à nouveau d'exécuter la tâche à $st+7$. Entre $st+3$ et $st+5$, l'agent est donc en situation d'attente. Soit $\{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}$ la succession des dates de début pour la tâche t_j dictées par la politique de l'agent. Dans notre exemple de la figure A.1, cet ensemble est défini par $\{st', st+5, st+7\}$.

Ainsi, nous pouvons décomposer $P^{st'}_{FAIL}$ de la façon suivante :

$$P^{st'}_{FAIL} = P^{st'}_{DM} + \sum_{t \in \{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} \left(\prod_{t' \leq t^-} P^{t'}_{PCV} \right) (P^{t}_{DM} - P^{t^-}_{DM}) + P^{UB}_{TL}$$

Lorsque l'agent essaye d'exécuter la tâche t_j à t, nous désignons par t^- la précédente date de début à laquelle l'agent a essayé d'exécuter la tâche. Par exemple, sur la figure A.1, si $t = st+7$ alors $t^- = st+5$.

$P^{st'}_{DM}$ peut par ailleurs être décomposée de la manière suivante :

$$P^{st'}_{DM} = P^{st}_{DM} + \sum_{t \in]st, st'[} (P^{t}_{DM} - P^{t-1}_{DM})$$

$P^{st'}_{fail}$ peut alors être définie par :

$$P^{st'}_{FAIL} = \overbrace{P^{st}_{fail} + \sum_{t \in]st, st'[} (P^{t}_{DM} - P^{t-1}_{fail})}^{A_2} + \overbrace{(P^{st'}_{DM} - P^{st'-1}_{DM})}^{B_2} + \overbrace{\sum_{t \in \{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} \left(\prod_{t' \leq t^-} P^{t'}_{PCV} \right) (P^{t}_{DM} - P^{t^-}_{DM})}^{C_2} + P^{UB}_{TL}$$

$P^{t'}_{PCV} \leq 1 \ \forall t'$ donc $\prod_{t' \leq st'} P^{t'}_{PCV} \leq 1$

Nous en déduisons alors que : $A_1 \leq A_2$, $B_1 \leq B_2$

Montrons à présent que : $C_1 \leq C_2$

$$
\begin{aligned}
C_2 &= \sum_{t\in\{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} (\prod_{t'\leq t^-} P^{t'}_{PCV})(P^t_{DM} - P^{t^-}_{DM}) \\
&= \sum_{t\in\{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} (\prod_{t'\leq t^-} P^{t'}_{PCV})(P^t_{DM} - P^{t-1}_{DM} + P^{t-1}_{DM} - P^{t-2}_{DM} + P^{t-2}_{DM} + \cdots - P^{t^-}_{DM}) \\
&= \sum_{t\in\{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} \Big((\prod_{t'\leq t} P^{t'}_{PCV})(P^t_{DM} - P^{t-1}_{DM}) + (\prod_{t'\leq t^-} P^{t'}_{PCV})(P^{t-1}_{DM} - P^{t-2}_{DM}) \\
&+ \cdots + (\prod_{t'\leq t} P^{t'}_{PCV})(P^{t^-+1}_{DM} - P^{t^-}_{DM})\Big)
\end{aligned}
$$

$$
\begin{aligned}
C_1 &= \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV})(P^t_{DM} - P^{t-1}_{DM}) \\
&= \sum_{t\in\{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} \Big((\prod_{t'<t} P^{t'}_{PCV})(P^t_{DM} - P^{t-1}_{DM}) + (\prod_{t'<t-1} P^{t'}_{PCV})(P^{t-1}_{DM} - P^{t-2}_{DM}) \\
&+ \cdots + (\prod_{t'<t^-+1} P^{t'}_{PCV})(P^{t^-+1}_{DM} - P^{t^-}_{DM})\Big)
\end{aligned}
$$

$P^{t'}_{PCV} \leq 1\ \forall t'$ et $\prod_{t'<t-i} P^{t'}_{PCV} = \prod_{t'\leq t^-} P^{t'}_{PCV} \prod_{t'\in]t^-, t-i[} P^{t'}_{PCV}$
donc $\prod_{t'<t-i} P^{t'}_{PCV} \geq \prod_{t'<t^-} P^{t'}_{PCV}\ \forall t'$

Ainsi, $C1 \leq C2$. Nous avons déjà montré que $A_1 \leq A_2$ et $B_1 \leq B_2$

Nous en déduisons que : $P^{st'}_{FAIL} \geq P^{st}_{FAIL}$

Les agents peuvent soit échouer soit réussir l'exécution de la tâche t_j, donc $P^i_{SUC} + P^i_{FAIL} = 1$ $\forall i$ où P^i_{SUC} désigne la probabilité que l'exécution de t_j réussisse.

Nous pouvons en déduire que :

$$
\begin{aligned}
&P^{st'}_{SUC} \leq P^{st}_{SUC} \\
&P^{st'}_{FAIL} \geq P^{st}_{FAIL} \\
&P^{st'}_{FAIL} R_{t_j} \geq P^{st}_{FAIL} R_{t_j} \\
&-P^{st'}_{FAIL} R_{t_j} \leq -P^{st}_{FAIL} R_{t_j} \\
&P^{st'}_{SUC} R_{t_j} \leq P^{st}_{SUC} R_{t_j} \\
&-P^{st'}_{FAIL} R_{t_j} + P^{st'}_{SUC} R_{t_j} \leq -P^{st}_{FAIL} R_{t_j} + P^{st}_{SUC} R_{t_j} \\
&V_{st'} \leq V_{st}
\end{aligned}
$$

Si l'agent suit la politique EST, il maximise donc son utilité espérée.

2) Pour toute tâche t_j exécutée par un agent $\mathcal{A}g_j$:

Soit t_{j+1} la tâche devant être exécutée par l'agent $\mathcal{A}g_j$ après t_j. Supposons que l'agent suive la politique EST après avoir exécuté la tâche t_j. De ce fait, l'agent commencera t_{j+1} au plus tôt. Soit st_{j+1} la première date de début possible pour la tâche t_{j+1} et st'_{j+1} une date de début de

t_{j+1} telle que : $st'_{j+1} > st_{j+1}$. Par hypothèse : $V_{st_{j+1}} \geq V_{st'_{j+1}}$. Nous allons démontrer que cette hypothèse est également vraie pour t_j. Soit st la première date de début possible pour l'exécution de t_j et st' une date de début de t_j telle que $st' > st$.

Comme nous l'avons précédemment démontré,

$$
\begin{aligned}
V_{st} =& V^{st}_{SUC} + V^{st}_{FAIL} \\
V^{st}_{FAIL} =& P^{st}_{FAIL} \cdot V^{st}_{fail} = P^{st}_{FAIL} \cdot V([failure_{t_j}, *, *]) \forall st \\
V^{st}_{SUC} =& P^{st}_{suc} V^{st}_{suc} + \sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{t}_{suc} + (\prod_{t'<st'} P^{t'}_{PCV}) P^{st'}_{suc} V^{st'}_{suc} + \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{t}_{suc}
\end{aligned}
$$

Par ailleurs,

$$V^{st}_{suc} = V_{st_{j+1}} \text{et } V^{st'}_{suc} = V_{st'_{j+1}}$$

Par hypothèse, $V^{st}_{suc} \geq V^{st'}_{suc}$
Donc,

$$
\begin{aligned}
V^{st}_{SUC} \geq& P^{st}_{suc} V^{st'}_{suc} + \sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{st'}_{suc} + (\prod_{t'<st'} P^{t'}_{PCV}) P^{st'}_{suc} V^{st'}_{suc} \\
&+ \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{t}_{suc} \\
V^{st}_{SUC} \geq& \Big(P^{st}_{suc} + \sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} + (\prod_{t'<st'} P^{t'}_{PCV}) P^{st'}_{suc} \Big) \cdot V^{st'}_{suc} \\
&+ \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} V^{t}_{suc} \\
V^{st'}_{SUC} =& P^{st'}_{suc} V^{st'}_{suc} + \sum_{t \in \{st'_k, st'_{k+j}, \cdots, st'_{k+j+p}\}} (\prod_{t' \leq t^-} P^{t'}_{PCV}) \cdot P^{t}_{suc} V^{t}_{suc} \\
P^{st}_{SUC} =& P^{st}_{suc} + \sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} + (\prod_{t'<st'} P^{t'}_{PCV}) P^{st'}_{suc} + \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV} P^{t}_{suc}) \\
P^{st}_{SUC} =& P^{st'}_{suc} + \sum_{t>st'} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc}
\end{aligned}
$$

Rappelons que :

$$P^{st}_{SUC} + P^{st}_{FAIL} = 1 \text{ et } P^{st'}_{SUC} + P^{st'}_{FAIL} = 1$$

Il peut être montré que $P^{st'}_{FAIL} \geq P^{st}_{FAIL}$ (voir étape précédente). Nous en déduisons que :

$$P^{st}_{suc} + \sum_{t \in]st, st'[} (\prod_{t'<t} P^{t'}_{PCV}) P^{t}_{suc} + (\prod_{t'<st'} P^{t'}_{PCV}) P^{st'}_{suc} \geq P^{st'}_{suc}$$

et $V^{st}_{SUC} \geq V^{st'}_{SUC}$

$$
\begin{aligned}
V_{st} &= V_{SUC}^{st} + V_{FAIL}^{st} \\
V_{st} &\geq V_{SUC}^{st'} + P_{FAIL}^{st} \cdot V_{fail}^{st} = V_{SUC}^{st'} + P_{FAIL}^{st} \cdot V_{fail}^{st'} \\
V_{st} &\geq V_{SUC}^{st'} + P_{FAIL}^{st'} \cdot V_{fail}^{st'} \\
V_{st} &\geq V_{SUC}^{st'} + V_{FAIL}^{st'} = V_{st'}
\end{aligned}
$$

Nous pouvons conclure que : $V_{st'} \leq V_{st}$.

Si l'agent suit la politique EST, il maximise son utilité espérée.

Nous avons montré que quelle que soit la tâche qu'un agent doit exécuter, s'il suit la politique EST, il maximisera son utilité espérée. Cette politique est donc un politique individuelle optimale.

Les agents doivent cependant agir de façon coopérative et pour ce faire ils doivent maximiser la somme de leurs utilités espérées (utilité espérée jointe). Si chaque agent suit la politique EST, la somme des utilités espérées est maximisée. Néanmoins, cette configuration n'est envisageable que si aucun agent empêche les autres agents de suivre la politique EST. Si un agent suivant la politique EST empêche un autre agent de suivre cette politique, alors l'utilité espérée jointe ne sera pas maximisée.

Considérons deux agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$ devant respectivement exécuter les tâches t_i et t_j. Nous supposerons que l'exécution de t_i doit être terminée pour que t_j puisse commencer. Si l'agent $\mathcal{A}g_i$ suit la politique EST, il commence l'exécution de t_i le plus tôt possible et il finit par conséquent le plus tôt possible. Ainsi, l'agent $\mathcal{A}g_j$ n'est pas retardé par $\mathcal{A}g_i$ et il peut également suivre la politique EST. Les deux agents maximisent alors leur utilité espérée. Si l'agent $\mathcal{A}g_i$ retarde l'exécution de t_i, son utilité espérée diminuera et il retardera l'exécution de t_j. L'agent $\mathcal{A}g_j$ ne pourra alors commencer au plus tôt et son utilité espérée sera également réduite. Par conséquent, l'utilité espérée jointe diminuera. Nous pouvons en conclure que si tous les agents suivent la politique EST, ils maximiseront la somme de leurs utilités espérées. La politique EST est donc une politique jointe optimale.□

A.2.3 Preuve du théorème 3

Si aucune contrainte temporelle n'est imposée sur les dates de fin des tâches ($LET \simeq +\infty$) alors, la politique LST est une politique optimale jointe.

Preuve :

Si aucune contrainte temporelle n'est imposée sur les dates de fin des tâches, alors quelle que soit la politique de l'agent, celui-ci respectera toujours les contraintes temporelles. Les cas d'échec total pour violation de la date de fin au plus tard n'ont pas à être envisagés.

La probabilité P_{DM} du système de transition peut donc être omise. L'utilité espérée est alors calculée de la manière suivante :

$$V' = V_{suc} + V_{PCV} + V_{LR} + V_{TL}$$

Rappelons que le but des agents est de maximiser la somme de leurs utilités espérées. Soit π_i la politique locale d'un agent $\mathcal{A}g_i$. La politique optimale pour les agents est la politique jointe π^* telle que :

$$\pi^* = arg\_max_{\pi=\langle\pi_1,\pi_2,\cdots,\pi_n\rangle} \sum_i V^{\pi_i}$$

où V^{π_i} correspond à l'utilité espérée de l'agent $\mathcal{A}g_i$ lorsqu'il suit la politique π_i.

Tout d'abord, nous montrerons que sans contraintes temporelles sur les dates de fin, la politique LST maximise l'utilité espérée d'un agent. Elle correspond donc à une politique individuellement optimale. Nous raisonnerons récursivement. Nous souhaitons démontrer que si un agent commence l'exécution d'une tâche au plus tard, son utilité est maximum. Nous montrerons que cette hypothèse est vraie pour la dernière tâche d'un agent. Puis, nous supposerons, par hypothèse de récurrence, que l'hypothèse est vraie pour la $n^{\text{ième}}$ tâche, et nous prouverons que l'hypothèse se vérifie alors pour la $n-1^{\text{ième}}$ tâche.

Soit $\mathcal{A}g_j$ un agent devant exécuter une tâche t_j. Soient r la quantité de ressources disponibles par l'agent $\mathcal{A}g_j$ et st la dernière date de début possible de t_j (date de début dictée par la politique LST). Soit st' une date de début pour t_j telle que $st' < st$. $V_{st,r}$ désigne l'utilité espérée de l'agent lorsqu'il commence l'exécution de la tâche à st avec r ressources. Nous souhaitons montrer que :

$$V_{st,r} \geq V_{st',r} \ \forall st' \ \forall r$$

Lorsque l'agent commence l'exécution de t_j à st avec r ressources, il maximise la probabilité que les prédécesseurs aient terminé. Comme il n'existe pas de contraintes temporelles sur les dates de fin, l'agent n'augmente pas la probabilité de violer la date de fin au plus tard (cette probabilité est nulle). L'agent ne peut de plus pas échouer partiellement puisque st est la dernière date de début possible de la tâche, tout échec est donc total.

Étant donné qu'il n'existe pas de contraintes sur les dates de fin des tâches, la dernière date de début st de t_j correspond par ailleurs à la dernière date de fin possible des prédécesseurs. Si les prédécesseurs n'ont pas terminé à st, ils ne termineront jamais (ils peuvent avoir échoué totalement par manque de ressources).

En revanche, si l'agent commence l'exécution de t_j à st' avec r ressources, la probabilité que les prédécesseurs aient terminé est plus faible qu'à st. La probabilité que l'agent échoue partiellement est donc plus importante. Un échec partiel consommant des ressources, l'agent augmente alors les risques de manquer de ressources et donc d'arriver dans un état d'échec total.

Si $\mathcal{A}g_j$ commence à exécuter la tâche t_j à st, l'exécution peut réussir ou échouer totalement.

$$V_{st,r} = P_{suc}^{st,r} V_{suc}^{st,r} + P_{fail}^{st,r} V_{fail}^{st,r}$$

où $P_{fail}^{st,r} = P_{LR}^{st,r} + P_{TL}^{st,r}$ et $V_{fail}^{st,r} = V([failure_{t_j}, *, *])$

$V_{suc}^{st,r}$ dépend de la durée d'exécution de la tâche. $P_{suc}^{st,r}$ correspond à la probabilité que l'exécution de la tâche se termine avec succès en commençant à st et $P_{fail}^{st,r}$ désigne la probabilité que l'exécution de la tâche échoue totalement en commençant à st (soit par manque de ressources (LR), soit parce que les prédécesseurs n'ont pas terminé (TL)).

De la même façon :

$$V_{st',r} = P_{suc}^{st',r} V_{suc}^{st',r} + P_{PCV}^{st',r} V_{PCV}^{st',r} + P_{fail}^{st',r} V_{fail}^{st',r}$$

Comme nous l'avons déjà prouvé dans une démonstration précédente (théorème 2 sur l'optimalité de la politique EST sous ressources illimitées), $V_{st',r}$ peut être décomposée de la façon suivante :

$$V_{st',r} = V_{SUC}^{st',r} + V_{FAIL}^{st',r}$$

où $V_{SUC}^{st',r}$ regroupe toutes les exécutions de t_j réalisées avec succès et $V_{FAIL}^{st',r}$ regroupe les exécutions ayant menées à un échec total. De plus,

$$V_{FAIL}^{st',r} = P_{FAIL}^{st',r} \cdot V_{fail}^{st',*}$$

où $P_{FAIL}^{st',r}$ désigne la probabilité que l'exécution de t_j échoue totalement.

Pour des raison de clarté, nous supposerons que toutes les dates appartenant à l'intervalle $[st', UB_j]$ sont des dates de début possibles pour t_j. Cependant, ces équations peuvent être adaptées afin de prendre en compte des cas où certaines dates appartenant à $[st', UB_j]$ ne sont pas des dates de début possibles pour t_j.

Traitons le cas où l'agent commence l'exécution de t_j à st' et échoue partiellement. Il choisit alors une date de début appartenant à l'intervalle $[st' + 1, UB_j = st]$. Nous noterons par t les dates de début sélectionnées ainsi par l'agent.

$V_{SUC}^{st',r}$ peut être décomposée de la manière suivante :

$$V_{SUC}^{st',r} = P_{suc}^{st',r} V_{suc}^{st',r} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r'} V_{suc}^{t,r'} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r'} V_{suc}^{st,r'}$$

où t' et t désignent des dates de début de t_j

Il est important de noter que les échec partiels consomment des ressources, c'est pourquoi nous utilisons r' pour désigner le taux de ressources disponibles en cas d'échec partiel. $r' = r - nbEP \cdot \Delta_r'$

où $nbEP$ est le nombre d'échecs partiels ayant eu lieu entre st' et t ; il dépend de la politique suivie par l'agent.

De même,

$$P_{FAIL}^{st',r} = P_{LR}^{st',r} + \sum_{t\in]st',st[} (\prod_{t'<t} P_{PCV}^{t',*}) \cdot (P_{LR}^{t,r''} - P_{LR}^{t-1,r'}) + (\prod_{t'<st} P_{PCV}^{t',r'}) \cdot (P_{LR}^{st,r''} - P_{LR}^{st-1,r'}) + P_{TL}^{st,r''}$$

où $st-1$ est la dernière date de début à laquelle on a tenté l'exécution de t_j avant d'arriver à st.

r'' est le taux de ressources restant lorsque l'agent a échoué partiellement et que ces échecs partiels l'ont mené à t. De même r' correspond au taux de ressources restant lorsque l'agent a échoué partiellement et que ces échecs partiels l'ont mené à $t-1$. Étant donné que moins d'échecs partiels ont eu lieu à $t-1$ qu'à t, nous avons : $r > r' > r''$.

De manière générale, moins un agent a de ressources, plus il a de chances d'échouer par manque de ressources : $\forall st, r, r'$, si $r < r'$ alors $P_{fail}^{st,r} \geq P_{fail}^{st,r'}$.

De même, $\forall st, st', r, r'$, si $r < r'$ alors

$$P_{LR}^{st,r} < P_{LR}^{st',r'} \tag{A.24}$$

Pour tout date de début t telle que $t \leq UB_j$, $P_{TL}^{st,r} = 0$ d'où :

$$P_{fail}^{st,r} = P_{LR}^{st,r}$$

A partir de l'équation A.24, nous pouvons déduire :
$P_{LR}^{t,r''} - P_{LR}^{t-1,r'} > 0$ et $P_{LR}^{st,r''} - P_{LR}^{st-1,r'} > 0$

De plus, $P_{LR}^{st',r} = P_{LR}^{st,r}$ et $P_{TL}^{st,r''} = P_{TL}^{st,r}$ $\forall r, r'$

Par ailleurs,
$P_{FAIL}^{st,r} = P_{fail}^{st,r} = P_{LR}^{st,r} + P_{TL}^{st,r}$

Nous obtenons : $P_{FAIL}^{st',r} \geq P_{FAIL}^{st,r}$

L'agent réussit ou échoue donc $P_{SUC}^{i,r} + P_{FAIL}^{i,r} = 1$ $\forall i$ $\forall r$.

Nous en déduisons :

$$P_{SUC}^{st',r} \leq P_{SUC}^{st,r}$$

1) Si t_j est la dernière tâche de $\mathcal{A}g_j$:

Nous avons alors : $V_{suc}^{st,r} = R_{t_j}\ \forall st\ \forall r$ et $V_{fail}^{st,r} = -R_{t_j}\ \forall st\ \forall r$.

$$\begin{aligned}
&P_{FAIL}^{st',r} \geq P_{FAIL}^{st,r}\\
&P_{FAIL}^{st',r} \cdot R_{t_j} \geq P_{FAIL}^{st,r} \cdot R_{t_j}\\
&-P_{FAIL}^{st',r} \cdot R_{t_j} \leq -P_{FAIL}^{st,r} \cdot R_{t_j}\\
&P_{SUC}^{st',r} \cdot R_{t_j} \leq P_{SUC}^{st,r} \cdot R_{t_j}\\
&-P_{FAIL}^{st',r} \cdot R_{t_j} + P_{SUC}^{st',r} \cdot R_{t_j} \leq -P_{FAIL}^{st,r} \cdot R_{t_j} + P_{SUC}^{st,r} \cdot R_{t_j}\\
&V_{st',r} \leq V_{st,r}
\end{aligned}$$

Si l'agent suit la politique LST, il maximise son utilité espérée.

2) Si t_j n'est pas la dernière tâche de $\mathcal{A}g_j$:

Par hypothèse de récurrence, si $st' < st$ alors $V_{suc}^{st',r} \leq V_{suc}^{st',r}$.

Par ailleurs,

$V_{fail}^{st,r} = V_{fail}^{st',r'}\ \forall$ st, st ', r, r'

$V_{SUC}^{st,r} = P_{suc}^{st,r} V_{suc}^{st,r}$

Nous pouvons alors décomposer $V_{SUC}^{st',r}$ de la manière suivante :

$$V_{SUC}^{st',r} = P_{suc}^{st',r} V_{suc}^{st',r} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r'} V_{suc}^{t,r'} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r''} V_{suc}^{st,r''}$$

tel que $r > r' > r''$

Proposition 1 *Soient r et r' deux taux de ressources tels que $r > r'$, alors $V_{st',r} \geq V_{st',r'}$.*

Preuve de la Proposition : cf. fin de la démonstration

D'après la proposition précédente, si r et r' sont deux taux de ressources tels que $r > r'$, alors $V_{suc}^{st',r} \geq V_{suc}^{st',r'}$. D'où :

$$\begin{aligned}
V_{SUC}^{st',r} &\leq P_{suc}^{st',r} V_{suc}^{st',r} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r'} V_{suc}^{st',r} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r''} V_{suc}^{st',r}\\
V_{SUC}^{st',r} &\leq \Big(P_{suc}^{st',r} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r'} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r''} \Big) V_{suc}^{st',r}\\
V_{SUC}^{st',r} &\leq P_{SUC}^{st',r} V_{suc}^{st',r} \qquad (A.25)
\end{aligned}$$

Par ailleurs,

$$V_{st,r} = P_{SUC}^{st,r} V_{suc}^{st,r} + P_{FAIL}^{st,r} V_{fail}^{st,r}$$

et

$$V_{st',r} = V_{SUC}^{st',r} + P_{FAIL}^{st',r} V_{fail}^{st',r}$$

Étant donné que $P^{st,r}_{FAIL} \geq P^{st',r}_{FAIL}$, nous pouvons ré-écrire $V_{st,r}$ de la manière suivante :

$$V_{st,r} = P^{st,r}_{suc} V^{st,r}_{suc} + P^{st,r}_{FAIL} V^{st,r}_{fail} = P^{st,r}_{suc1} V^{st,r}_{suc} + P^{st,r}_{suc2} V^{st,r}_{suc} + P^{st,r}_{FAIL} V^{st,r}_{fail} \tag{A.26}$$

tel que $P^{st,r}_{suc1} + P^{st,r}_{suc2} = P^{st,r}_{suc}$ et $P^{st,r}_{suc2} = P^{st,r}_{FAIL} - P^{st',r}_{FAIL} = P^{st,r}_{FAIL} - P^{st',r}_{FAIL}$
D'après l'équation A.25, nous déduisons que :

$$V_{st',r} \leq P^{st',r}_{SUC} V^{st',r}_{suc} + P^{st',r}_{fail} V^{st',r}_{fail}$$

Par hypothèse de récurrence : $V^{st,r}_{suc} \geq V^{st',r}_{suc}$
Donc

$$P^{st',r}_{SUC} V^{st,r}_{suc} \geq P^{st',r}_{SUC} V^{st',r}_{suc} \tag{A.27}$$

D'autre part, $\forall st, r$:

$$V^{st,r}_{suc} > V^{st,*}_{fail}$$

D'où

$$P^{st,r}_{suc2} V^{st,r}_{suc} > P^{st,r}_{suc2} V^{st,*}_{fail}$$

$$P^{st,r}_{suc2} V^{st,r}_{suc} + P^{st,r}_{FAIL} V^{st,*}_{fail} > P^{st,r}_{suc2} V^{st,*}_{fail} + P^{st,r}_{FAIL} V^{st,*}_{fail}$$

$$\begin{aligned} P^{st,r}_{suc2} V^{st,*}_{fail} + P^{st,r}_{FAIL} V^{st,r}_{fail} &= \left(P^{st,r}_{suc2} + P^{st,r}_{FAIL}\right) V^{st,*}_{fail} \\ &= P^{st',r}_{FAIL} V^{st,*}_{fail} \\ &= P^{st',r}_{FAIL} V^{st',*}_{fail} \end{aligned}$$

Nous en déduisons :

$$P^{st,r}_{suc2} V^{st,r}_{suc} + P^{st,r}_{FAIL} V^{st,*}_{fail} > P^{st',r}_{FAIL} V^{st',*}_{fail} \tag{A.28}$$

D'après les équations A.27 et A.28,

$$P^{st',r}_{SUC} V^{st',r}_{suc} + P^{st',r}_{FAIL} V^{st',*}_{fail} \leq P^{st',r}_{SUC} V^{st,r}_{suc} + P^{st,r}_{suc2} V^{st,r}_{suc} + P^{st,r}_{FAIL} V^{st,*}_{fail}$$

De plus, $P^{st',r}_{SUC} = P^{st,r}_{suc1}$
D'où

$$P^{st',r}_{SUC} V^{st',r}_{suc} + P^{st',r}_{FAIL} V^{st',*}_{fail} \leq P^{st,r}_{suc1} V^{st,r}_{suc} + P^{st,r}_{suc2} V^{st,r}_{suc} + P^{st,r}_{FAIL} V^{st,*}_{fail}$$

et $V_{st,r} \geq V_{st',r}$.

Si un agent suit la politique LST, il maximise sa propre utilité espérée.

Nous avons montré que quelle que soit la tâche t_i d'un agent, si ce dernier commence l'exécution de t_i au plus tard, il maximise son utilité espérée. La politique LST est donc une politique individuellement optimale. Les agents doivent cependant agir de manière coopérative, ils cherchent donc à maximiser la somme de leurs utilités et non leur propre utilité. Si chaque agent suit la politique LST, la somme des utilités sera maximale. Cependant, le fait qu'un agent suive la politique LST ne doit pas empêcher un autre agent de suivre cette politique.

Considérons deux agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$ devant exécuter respectivement les tâches t_i et t_j. Nous supposons qu'il existe une contrainte de précédence entre t_i et t_j : t_i doit être terminée pour que t_j puisse commencer. Si l'agent $\mathcal{A}g_i$ suit la politique LST, il commence l'exécution de la tâche au plus tard et la termine, suivant la durée d'exécution, au plus tard. En terminant l'exécution de la tâche t_i au plus tard, l'agent $\mathcal{A}g_i$ n'empêche pas $\mathcal{A}g_j$ de commencer l'exécution de t_j au plus tard. En effet, comme il n'y a pas de contraintes temporelles sur les dates de fin des tâches, la dernière date de début de t_j sera toujours une date de début possible, même si t_i finit au plus tard (cf. algorithme de propagation temporelle). Comme le fait qu'un agent suive sa politique LST n'empêche pas les autres agents de suivre LST, les agents peuvent tous maximiser leurs utilités et ainsi maximiser la somme des utilités. Les agents adoptent ainsi une politique optimale jointe.□

Preuve de la Proposition 1

Moins l'agent a des ressources, plus la probabilité qu'il manque de ressources augmente donc : $P_{LR}^{st',r'} \leq P_{LR}^{st',r}$.

Les probabilités des autres transitions d'échec total ne varient pas (probabilités que les prédécesseurs n'aient pas terminé). Donc $P_{FAIL}^{st',r'} \leq P_{FAIL}^{st',r}$ et par conséquent $P_{SUC}^{st',r'} \geq P_{SUC}^{st',r}$

1) Si t_j est la dernière tâche de l'agent $V_{suc}^{st',r} = R_{t_j}\ \forall st, \forall r$ et $V_{fail}^{st',r} = -R_{t_j}\ \forall st, \forall r$

$$
\begin{aligned}
&P_{FAIL}^{st',r} \geq P_{FAIL}^{st',r'} \\
&P_{FAIL}^{st',r} \cdot R_{t_j} \geq P_{FAIL}^{st',r'} \cdot R_{t_j} \\
&-P_{FAIL}^{st',r} \cdot R_{t_j} \leq -P_{FAIL}^{st',r'} \cdot R_{t_j} \\
&P_{SUC}^{st',r} \cdot R_{t_j} \leq P_{SUC}^{st',r'} \cdot R_{t_j} \\
&-P_{FAIL}^{st',r} \cdot R_{t_j} + P_{SUC}^{st',r} \cdot R_{t_j} \leq -P_{FAIL}^{st',r'} \cdot R_{t_j} + P_{SUC}^{st',r'} \cdot R_{t_j} \\
&V_{st',r} \leq V_{st',r'}
\end{aligned}
$$

La proposition est donc vraie pour la dernière tâche de l'agent.

2) Si t_j n'est pas la dernière tâche de l'agent :

Par hypothèse de récurrence, si $r' < r$ alors $V_{suc}^{st',r'} \leq V_{suc}^{st',r}$.

$$P_{FAIL}^{st',r} \geq P_{FAIL}^{st',r'}$$

Nous pouvons décomposer $V_{SUC}^{st',r'}$ de la façon suivante :

$$V_{SUC}^{st',r'} = P_{suc}^{st',r'} V_{suc}^{st',r'} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r1'} V_{suc}^{t,r1'} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r1''} V_{suc}^{st,r1''}$$

tel que $r' > r1' > r1''$ où $r1'$ et $r1''$ sont calculés en fonction du nombre d'échecs partiels ayant eu lieu.

$$V_{SUC}^{st',r} = P_{suc}^{st',r} V_{suc}^{st',r} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r1} V_{suc}^{t,r2'} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r2''} V_{suc}^{st,r2''}$$

tel que $r > r2' > r2''$ où $r2'$ et $r2''$ sont calculés en fonction du nombre d'échecs partiels ayant eu lieu.

Si $r > r'$ alors $r2' > r1'$ ($r2' = r - NBEP \cdot rVerif$ et $r1' = r' - NBEP \cdot rVerif$) et $r2'' > r1''$

Par hypothèse de récurrence :

$V_{suc}^{st',r'} \leq V_{suc}^{st',r}$

$V_{suc}^{st',r1'} \leq V_{suc}^{st',r2'}$

$V_{suc}^{st',r1''} \leq V_{suc}^{st',r2''}$

D'où

$$\begin{aligned}
V_{SUC}^{st',r'} \leq & P_{suc}^{st',r'} V_{suc}^{st',r} + \sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) P_{suc}^{t,r1'} V_{suc}^{t,r2'} + (\prod_{t'<st} P_{PCV}^{t'}) P_{suc}^{st,r1''} V_{suc}^{st,r2''} \\
P_{suc}^{st',r} = & P_{suc}^{st',r'} + (P_{LR}^{st',r'} - P_{LR}^{st',r}) \\
P_{suc}^{t,r2'} = & P_{suc}^{t,r1'} + (P_{LR}^{t,r1'} - P_{LR}^{t,r2'}) \\
P_{suc}^{st,r2'} = & P_{suc}^{st,r1''} + (P_{LR}^{st,r1''} - P_{LR}^{st,r2''})
\end{aligned}$$

D'où

$$\begin{aligned}
V_{st',r'} \leq & \overbrace{P_{suc}^{st',r'} V_{suc}^{st',r} + P_{LR}^{st',r'} V_{fail}^{st',*}}^{A1} + \overbrace{\sum_{t \in]st',st[} (\prod_{t'<t} P_{PCV}^{t'}) \Big(P_{suc}^{t,r1'} V_{suc}^{t,r2'} + P_{LR}^{t',r1'} V_{fail}^{st',*} \Big)}^{B1} \\
& + \overbrace{(\prod_{t'<st} P_{PCV}^{t'}) \Big(P_{suc}^{st,r1''} V_{suc}^{st,r2''} + P_{LR}^{st,r1''} V_{fail}^{st',*} \Big)}^{C1} + P_{TL}^{st,r1''}
\end{aligned}$$

$$V_{st',r} = \overbrace{(P_{suc}^{st',r'} + (P_{LR}^{st',r'} - P_{LR}^{st',r}))V_{suc}^{st',r} + (P_{LR}^{st',r'} - (P_{suc}^{st',r} - P_{suc}^{st',r'}))V_{fail}^{st',*}}^{A2}$$
$$+ \overbrace{\sum_{t\in]st',st[\ t'<t} (\prod P_{PCV}^{t'})\Big((P_{suc}^{t,r1'} + (P_{LR}^{t,r1'} - P_{LR}^{t,r2'}))V_{suc}^{t,r2'} + (P_{LR}^{t',r1'} - (P_{suc}^{t,r2'} - P_{suc}^{t,r1'}))V_{fail}^{st',*}\Big)}^{B2}$$
$$+ \overbrace{(\prod_{t'<st} P_{PCV}^{t'})\Big((P_{suc}^{st,r1''} + (P_{LR}^{st,r1''} - P_{LR}^{st,r2''}))V_{suc}^{st,r2''} + (P_{LR}^{st,r1''} - (P_{suc}^{st,r2'} - P_{suc}^{st,r1''})V_{fail}^{st',*}\Big)}^{C2} + P_{TL}^{st,r1''}$$

$P_{suc}^{st',r'} \leq P_{suc}^{st',r}$ et $P_{LR}^{st',r'} \geq P_{LR}^{st',r}$

De plus, $V_{fail}^{st',*} < V_{suc}^{st',*}$, donc $A1 \leq A2$

De même, nous pouvons montrer que $B1 \leq B2$ et $C1 \leq C2$ (cf. démonstration du théorème 2), d'où $V_{st',r'} < V_{st',r}$. □

A.2.4 Preuve du théorème 4

Si les ressources sont illimitées et qu'aucune contrainte temporelle n'est imposée sur les dates de fin alors, toutes les politiques sont équivalentes et optimales.

Preuve :

Si les ressources sont illimitées et qu'aucune contrainte temporelle n'est imposée, alors les agents n'échoueront jamais totalement.

En effet, les tâches racines du graphe n'ont pas de prédécesseurs, la probabilité que les prédécesseurs aient terminé est donc de 1. Comme les ressources sont illimitées et qu'aucune contrainte temporelle n'est imposée, quelle que soit la date de début d'une tâche, l'exécution réussira. Soit une tâche t_j d'un niveau k. Nous supposons, par hypothèse de récurrence, que les tâches du niveau $k-1$ se sont toutes exécutées avec succès. Montrons que la tâche t_j n'échouera pas totalement.

- Les prédécesseurs de la tâche n'échoueront pas totalement donc $P_{TL} = 0\ \forall st_{t_j}$.
- Les ressources sont illimitées donc $P_{LR} = 0\ \forall st_{t_j}\ \forall r_{t_j}$.
- Les ressources sont illimitées donc $P_{DM} = 0\ \forall st_{t_j}$.

D'où $P_{fail}^{st_{t_j},*} = 0\ \forall st_{t_j}$. La tâche t_j n'échouera pas totalement.

Soit $\mathcal{A}g_j$ un agent devant exécuter la tâche t_j. Soir r la quantité de ressources disponibles par l'agent $\mathcal{A}g_j$. Soient st et st' deux dates de début pour t_j . $V_{st,r}$ désigne l'utilité espérée de l'agent lorsqu'il commence l'exécution de la tâche à st avec r ressources. Nous souhaitons montrer que $\forall st, st', r : V_{st,r} = V_{st',r}$

1) Si t_j est la dernière tâche de $\mathcal{A}g_j$:
t_j est la dernière tâche de l'agent d'où $V_{suc}^{st,r} = R_{t_j}\ \forall st\ \forall r$
et $V_{fail}^{st,r} = -R_{t_j}\ \forall st\ \forall r$.
$P_{fail}^{st_{t_j},*} = 0\ \forall st_{t_j}$
donc $P_{SUC}^{st_{t_j},r} = 1\ \forall st_{t_j}$
De plus, $V_{st,r} = P_{SUC}^{st,r} R_{t_j}$ et $V_{st',r} = P_{SUC}^{st',r} R_{t_j}$
D'où, $P_{SUC}^{st,r} = P_{SUC}^{st',r}$
Nous en concluons que : $V_{st,r} = V_{st',r}$.

2) Si t_j n'est pas la dernière tâche de $\mathcal{A}g_j$:
Par hypothèse de récurrence, $\forall st', st : V_{suc}^{st,r} = V_{suc}^{st',r}$.
$V_{st,r} = P_{SUC}^{st,r} V_{suc}^{st,r}$
$V_{st',r} = P_{SUC}^{st',r} V_{suc}^{st',r}$
De plus, $P_{SUC}^{st,r} = P_{SUC}^{st',r}$
D'où $V_{st,r} = V_{st',r}$.

Quelle que soit la date à laquelle un agent commence, son utilité espérée sera toujours la même, toutes les politiques π_i d'un agent $\mathcal{A}g_i$ sont donc individuellement optimales et toute combinaison de politiques individuelles résulte en une politique optimale jointe. □

A.2.5 Preuve du théorème 5

Si les ressources sont illimitées alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.

Preuve :

La politique EOC est calculée à partir de l'équation suivante :

$$\pi_i(s_i) = arg\_max_{E(t_{i+1},st), st \geq et(I')_{t_i}} \Big(\overbrace{V'}^{\text{Utilité espérée}} - \overbrace{OC(t_{i+1}, st)}^{\text{Coût occasionné}} \Big)$$

Soient t_{i+1} une tâche quelconque et st la première date de début possible pour t_{i+1}. Considérons st' une date de début de t_{i+1} telle que $st' > st$.

Nous avons précédemment démontré que $V_{st} \geq V_{st'}$.

Il peut de plus être démontré que $OC(t_{i+1}, st) \leq OC(t_{i+1}, st')$:

En effet, si les ressources sont illimitées, la propriété suivante est vérifiée : $OC_{t_j}(\Delta t, r) = OC_{t_j}(\Delta t, r')\ \forall r, r'$. De ce fait, les probabilités sur les taux de ressources et les consommations de ressources peuvent être omises.

$$\begin{aligned} OC_{t_j}(\Delta t, r) &= OC_{t_j}(\Delta t) \ \forall \ r \\ &= V_{t_j}^{0,r} - V_{t_j}^{\Delta t, r} \\ &= V_{st} - V_{st'} \end{aligned}$$

De même,

$$\begin{aligned} OC_{t_j}(\Delta t', r) &= OC_{t_j}(\Delta t') \ \forall \ r \\ &= V_{t_j}^{0,r} - V_{t_j}^{\Delta t', r} \\ &= V_{st} - V_{st''} \end{aligned}$$

Si $\Delta t' > \Delta t$ alors $st'' \geq st'$.

Nous pouvons affirmer que :

$$V_{st} \geq V_{st'} \geq V_{st''}$$

D'où :

$$OC_{t_j}(\Delta t, r) \geq OC_{t_j}(\Delta t', r)$$

Le coût occasionné espéré provoqué par un agent lorsqu'il finit à et_{t_i} est par ailleurs défini de la façon suivante :

$$\begin{aligned} EOC_{ag,t_i}(et_{t_i}) = & \sum_{st_{t_j} \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et_{t_i}, t_i) \cdot OC_{t_j}(st_{t_j} - LB, r_{t_j}) \\ & + (1 - (\sum_{st \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et_{t_i}, t_i))) \cdot OC_{t_j}(fail) \end{aligned}$$

De même,

$$\begin{aligned} EOC_{ag,t_i}(et'_{t_i}) = & \sum_{st_{t_j} \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et'_{t_i}, t_i) \cdot OC_{t_j}(st_{t_j} - LB, r_{t_j}) \\ & + (1 - (\sum_{st \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et'_{t_i}, t_i))) \cdot OC_{t_j}(fail) \end{aligned}$$

Si $et_{t_i} < et'_{t_i}$, le coût occasionné espéré provoqué par un agent lorsqu'il finit à et'_{t_i} est donc supérieur à celui provoqué lorsqu'il finit à et_{t_i}. En effet,

$$1 - (\sum_{st \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et'_{t_i}, t_i)) \geq 1 - (\sum_{st \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et_{t_i}, t_i))$$

De plus le retard provoqué sur un agent lorsque t_i finit à et_{t_i} est moins important que celui induit lorsque t_i s'achève à et'_{t_i}, d'où :

$$\sum_{st_{t_j} \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et_{t_i}, t_i) \cdot OC_{t_j}(st_{t_j} - LB, r_{t_j}) \leq \sum_{st_{t_j} \in ST(t_j)} P_{st}^{t_{i+1}}(st_{t_j}, et'_{t_i}, t_i) \cdot OC_{t_j}(st_{t_j} - LB, r_{t_j})$$

Nous en déduisons :

$$EOC_{ag,t_i}(et_{t_i}) \geq EOC_{ag,t_i}(et'_{t_i})$$

Afin de calculer le coût occasionné espéré provoqué par un agent lorsqu'il commence l'exécution de t_i à st, nous devons tenir compte des différentes issues possibles pour l'exécution de la tâche. Ainsi, chaque type de transitions est envisagé :

$$\begin{aligned} OC(t_{i+1}, st) =& P_{suc} \cdot \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(et_{i+1}) \\ &+ (P_{LR} + P_{TL} + P_{DM}) \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(fail) \\ &+ P_{PCV} \cdot OC(t_{i+1}, s = next\_start) \end{aligned}$$

$OC(t_{i+1}, st)$ peut être ré-écrit de la façon suivante :

$$OC(t_{i+1}, st) = OC^{st}_{SUC} + OC^{st}_{FAIL}$$

où :

$$OC^{st}_{SUC} = P_{suc} \cdot \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(et_{i+1}) + \sum_{t > st} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{suc} EOC_{ag,t_{i+1}}(et'_{i+1})$$

et

$$\begin{aligned} OC^{st}_{FAIL} =& P^{st}_{DM} \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(fail) \\ &+ \sum_{t > st} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{DM} \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(fail) \\ &+ P^{UB}_{TL} \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(fail) \\ OC^{st}_{FAIL} =& P^{st}_{FAIL} \cdot \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(fail) \end{aligned}$$

tel que

$$P^{st}_{FAIL} = P^{st}_{DM} + \sum_{t > st} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{DM} + P^{UB}_{TL}$$

Lors de la démonstration du théorème 2 nous avons démontré que pour tout st' tel que $st' > st$:

$$P^{st}_{FAIL} \leq P^{st'}_{FAIL}$$

Nous en déduisons que :

$$OC^{st}_{FAIL} \leq OC^{st'}_{FAIL}$$

Par ailleurs,

$$OC^{st}_{SUC} \leq OC^{st'}_{SUC}$$

En effet,

$$\begin{aligned} OC^{st}_{SUC} =& P^{st}_{suc} \cdot \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(et_{i+1}) + \sum_{t>st} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{suc} EOC_{ag,t_{i+1}}(et'_{i+1}) \\ =& P^{st}_{suc} \cdot \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(et_{i+1}) + \sum_{t|st'>t>st} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{suc} EOC_{ag,t_{i+1}}(et'_{i+1}) \\ &+ \sum_{t>st'} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{suc} EOC_{ag,t_{i+1}}(et'''_{i+1}) \end{aligned}$$

Et,

$$OC^{st'}_{SUC} = P^{st'}_{suc} \cdot \sum_{ag \in \text{other\_agents}} EOC_{ag,t_{i+1}}(et''_{i+1}) + \sum_{t>st'} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{suc} EOC_{ag,t_{i+1}}(et'''_{i+1})$$

où $et''_{i+1} \geq et_{i+1}$, $et'''_{i+1} \geq et'_{i+1}$ et

$$P^{st'}_{suc} = P^{st}_{suc} + \sum_{t|st'>t>st} (\prod_{t' \leq t} P^{t'}_{PCV}) P^{t}_{suc}$$

Si $et_{t_i} \leq et'_{t_i}$, nous avons démontré que :

$$EOC_{ag,t_i}(et_{t_i}) \geq EOC_{ag,t_i}(et'_{t_i})$$

De plus,

$$P^{st}_{suc} \geq P^{st'}_{suc}$$

Nous pouvons donc affirmer :

$$OC^{st}_{SUC} \leq OC^{st'}_{SUC}$$

Nous en déduisons :

$$OC(t_{i+1}, st) \leq OC(t_{i+1}, st')$$

D'où :

$$V'_{st} - OC(t_{i+1}, st) \geq V'_{st'} - OC(t_{i+1}, st')$$

La politique EOC dicte donc de commencer chaque tâche le plus tôt possible, elle est identique à la politique EST. Nous avons prouvé précédemment que lorsque les ressources sont illimitées, la politique EST est optimale. Nous en concluons que la politique EOC est optimale. □

A.2.6 Preuve du théorème 6

Si aucune contrainte temporelle n'est imposée sur les dates de fin des tâches ($LET \simeq +\infty$) alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.

Preuve :

La politique EOC est calculée à partir de l'équation suivante :

$$\pi_i(s_i) = arg\_max_{E(t_{i+1},st),st \geq et(I')_{t_i}} \Big(\overbrace{V'}^{\text{Utilité espérée}} - \overbrace{OC(t_{i+1},st)}^{\text{Coût occasionné}} \Big)$$

Soit t_{i+1} une tâche quelconque, st la dernière date de début possible de t_{i+1} et st' une date de début de t_{i+1} telle que $st' < st$.

Nous avons précédemment démontré (cf. démonstration du théorème 3) que $V_{st} \geq V_{st'}$.

La politique LST étant optimale, le coût occasionné est nul quels que soient le retard provoqué et les ressources disponibles. Nous en déduisons :

$$V'_{st} - OC(t_{i+1}, st) \geq V'_{st'} - OC(t_{i+1}, st')$$

La politique EOC dicte donc de commencer chaque tâche le plus tard possible, elle est identique à la politique LST. Nous avons prouvé précédemment que lorsqu'aucune contrainte temporelle n'est imposée, la politique LST est optimale. Nous en concluons que la politique EOC est optimale. □

A.2.7 Preuve du théorème 7

Si les ressources sont illimitées et qu'aucune contrainte temporelle n'est imposée sur les dates de fin alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.

Preuve :

D'après le théorème 4 toutes les politiques jointes sont équivalentes et optimales. La politique calculée par l'algorithme de révision des politiques est donc optimale. □

A.2.8 Preuve du théorème 8

Supposons que la politique initialement fixée soit la politique EST. Si le graphe de la mission ne comprend que deux agents $\mathcal{A}g_i$ et $\mathcal{A}g_j$ et que toutes les contraintes de précédence sont orientées de $\mathcal{A}g_i$ vers $\mathcal{A}g_j$ alors, l'algorithme de résolution d'un OC-DEC-MDP calcule une politique optimale.

Preuve :

Si les contraintes de précédence sont toutes orientées de $\mathcal{A}g_i$ vers $\mathcal{A}g_j$ alors, l'exécution des tâches de $\mathcal{A}g_i$ ne dépend pas de prédécesseurs exécutés par un autre agent. L'agent $\mathcal{A}g_i$ peut donc commencer chaque tâche au plus tôt sans risque de violer les contraintes de précédence. Nous pouvons alors démontrer que la politique EST permet à l'agent de maximiser sa propre utilité. Soit st la date de début au plus tôt d'une tâche t_{i+1} exécutée par $\mathcal{A}g_i$ et st' une autre date de

début ($st' > st$). Quelle que soit la date de début de t_{i+1}, la probabilité d'échouer partiellement est nulle. En revanche, plus l'agent retarde l'exécution de t_{i+1}, plus la probabilité de violer les contraintes temporelles est forte. Nous avons donc : $V_{st',r} \leq V_{st,r'}$ quelle que soit st'.

La politique EST maximise donc l'utilité espérée de l'agent $\mathcal{A}g_i$. Par ailleurs, plus l'agent $\mathcal{A}g_i$ retarde l'exécution de l'agent $\mathcal{A}g_j$, plus le coût occasionné provoqué sur ce dernier augmente. Nous avons en effet[57] :

$$OC(t_{i+1}, st) \leq OC(t_{i+1}, st')$$

La politique EOC est calculée à partir de l'équation suivante :

$$\pi_i(s_i) = arg\_max_{E(t_{i+1},st),st \geq et(I')_{t_i}} \Big(\overbrace{V'}^{\text{Utilité espérée}} - \overbrace{OC(t_{i+1}, st)}^{\text{Coût occasionné}} \Big)$$

Nous en déduisons que la politique EOC de l'agent $\mathcal{A}g_i$ lui dicte de commencer chaque tâche le plus tôt possible, elle est identique à la politique EST.

Si la politique initiale de l'agent $\mathcal{A}g_i$ est la politique EST, la politique de l'agent $\mathcal{A}g_i$ n'est pas modifiée lors de la résolution de l'OC-DEC-MDP. La fonction de transition de l'agent $\mathcal{A}g_j$ est donc déterminée à partir de la politique optimale de l'agent $\mathcal{A}g_i$. L'utilité espérée de l'agent est par conséquent calculée de manière précise en considérant que les autres agents suivent une politique optimale.

La politique de l'agent $\mathcal{A}g_j$ n'influence par ailleurs aucun autre agent. En effet, d'après nos hypothèses, il n'existe aucune contrainte de précédence entre l'agent $\mathcal{A}g_j$ et un autre agent. Le coût occasionné provoqué par $\mathcal{A}g_j$ est ainsi nul quel que soit ses décisions. La modification de la politique de l'agent $\mathcal{A}g_j$ n'aura donc aucun impact sur la politique de l'agent $\mathcal{A}g_i$.

La résolution de l'OC-DEC-MDP correspond donc au calcul de la politique de l'agent $\mathcal{A}g_j$ étant donnée la politique optimale de $\mathcal{A}g_i$ connue et fixée. Cette résolution conduit par conséquent au calcul de la politique optimale de l'agent $\mathcal{A}g_j$.

Lorsque l'algorithme itératif est utilisé, il est possible de considérer un quelconque ensemble de politiques initiales. La première itération détermine la politique optimale de l'agent $\mathcal{A}g_i$ (obtention de la politique EST). Nous obtenons alors le même ensemble de politiques que celui qui vient d'être considéré. La seconde itération détermine ensuite la politique optimale jointe en ne modifiant que la politique de l'agent $\mathcal{A}g_j$. Comme nous l'avons précédemment expliqué, le calcul de cette politique n'entraîne aucune modification de la politique de l'agent $\mathcal{A}g_i$. Aux itérations suivantes, la politique de cet agent ne changera pas et ce quelles que soient les modifications de la politique de l'agent $\mathcal{A}g_j$. Nous atteignons donc un équilibre dans lequel chaque agent suit

[57] Cette inégalité peut être démontrée en utilisant un raisonnement identique à celui effectué lors de la démonstration du théorème 5.

une politique individuelle optimale qui s'avère également correspondre à la politique de l'agent constituant la politique optimale jointe.

Annexe B

Présentation des robots Koala

Les robots Koala sont des robots mobiles développés par la société suisse K-Team. Ils s'inscrivent dans la continuation des robots Khepera de cette même société et proposent une plus grande diversité d'équipements que leurs « petits frères ».

Équipement de base des robots Koala

Les robots Koala sont principalement destinés à la recherche et au développement d'expérimentations en intérieur. Ces robots de 30cm sur 30cm sont équipés de différents éléments de base auxquels d'autres composants peuvent être ajoutés en fonction des besoins. Les Koala sont tous équipés d'un processeur Motorola 68331 à 22MHz, d'une quantité de mémoire RAM de 1 méga-octet et d'une mémoire ROM de 1 méga-octet (mémoire flash). Les robots sont par ailleurs alimentés par batteries offrant une autonomie de 2 à 6 heures. Deux moteurs (droit et gauche) actionnent les six roues des robots leur permettant ainsi de se déplacer à une vitesse allant de 0,005 m/s à 0,6m/s. Ces roues constituent les seuls effecteurs présents de base sur les Koala.

En ce qui concerne les capteurs, chaque robot est muni de 16 capteurs à infra-rouge positionnés tout autour de lui, dont une majorité (8 capteurs sur 16) se situe à l'avant (cf. figure B.1). Ces capteurs sont voués à la détection d'obstacles. Cette dernière est réalisée par mesure de la lumière ambiante et de la lumière réfléchie. Chaque capteur infrarouge est composé d'une diode émettant une lumière infrarouge et d'un récepteur permettant de mesurer la lumière ambiante. La lumière réfléchie est calculée en réalisant la différence entre la lumière mesurée lors de l'émission d'un rayon infra-rouge, et la lumière mesurée sans émission (lumière ambiante).

Les capteurs infrarouges retournent une mesure de lumière qui, quand elle est supérieure à un seuil donné, permet de déduire la présence d'un obstacle. Cette mesure de lumière est influencée par de nombreux facteurs parmi lesquels la texture, le matériau et la couleur de l'obstacle, les conditions d'éclairage, et l'angle entre le robot et l'obstacle. Suivant ces paramètres, l'obstacle sera détecté à plus ou moins longue distance. Une étude de l'influence de ces facteurs sur la

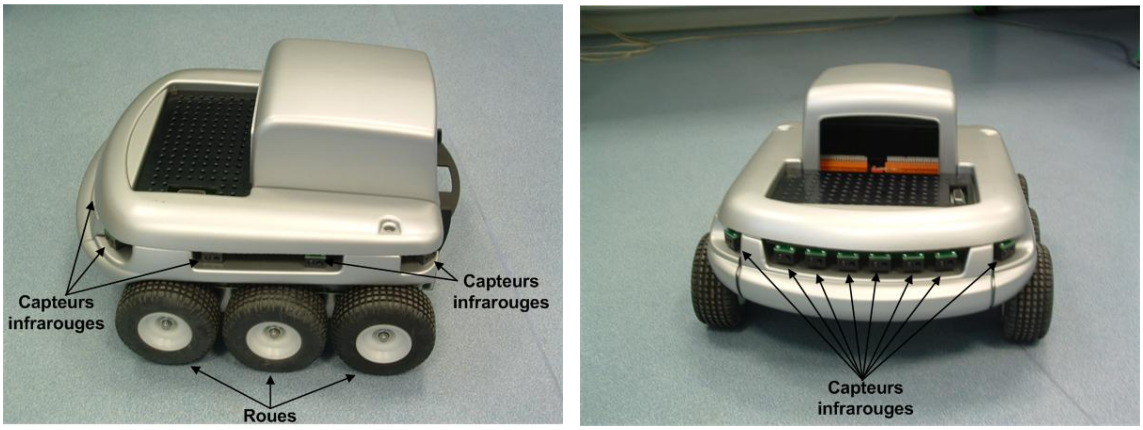


FIG. B.1 – Placement des capteurs sur les robots Koala

distance de détection est ébauchée dans le manuel utilisateur des robots Koala [K-Team, 2001]. La valeur de seuil permettant d'inférer la présence d'un obstacle correspond à une quantité de lumière, elle est fixée par l'utilisateur. Soit $\mathcal{S}_e$ une telle valeur de seuil. Si un obstacle est très réfléchissant, il renverra une forte lumière, la valeur $\mathcal{S}_e$ sera donc dépassée même si le robot est à longue distance de l'obstacle. Ce dernier sera donc détecté de loin. A l'inverse, un objet mat et sombre réfléchira peu de lumière et le robot devra en être très près pour que la valeur $\mathcal{S}_e$ soit dépassée et que l'obstacle soit détecté. Les capteurs ne mesurent donc pas la distance entre le robot et un obstacle mais ils permettent de déduire la présence d'un obstacle. En raison des nombreux facteurs influençant la valeur retournée par les capteurs infrarouges, il est généralement difficile d'établir la valeur de seuil adéquate. Il peut de plus être nécessaire, en raison des imprécisions des capteurs, d'établir des valeurs de seuil différentes pour chaque capteur.

Possibilité d'amélioration des capacités

Il est possible d'améliorer la précision des mesures et les capacités d'action des robots Koala en diversifiant les capteurs et les effecteurs qui les équipent. Ainsi des capteurs à ultra-sons ou à LED, une caméra, une boussole ou un GPS peuvent par exemple être ajoutés. Au niveau des effecteurs, il est possible de doter les Koala d'un bras muni d'une pince, d'un système d'aspiration, etc. Enfin, ces robots peuvent être équipés d'un système de communication sans fil permettant l'échange de messages entre les robots ou entre un robot et une machine. La commande à distance ou la déportation des calculs sur une machine distante sont alors envisageables.

Les robots Koala offrent également des possibilités d'ajout de mémoire et de puissance de calcul.

Spécifications des robots utilisés

Les expérimentations présentées dans la partie validation de ce document ont été réalisées avec des robots Koala munis des capteurs et effecteurs de base (roues et capteurs infrarouge) et n'ayant aucune possibilité de communiquer entre eux. La planification des tâches des robots a été réalisée sur une machine de bureau standard. Nous avons ensuite procédé, pour chaque robot, à un téléchargement en mémoire flash de sa politique et d'un interpréteur de politiques. Une fois cette manipulation réalisée, le robot est en mesure de commencer l'exécution de sa politique dès sa mise en marche.

Index

Bibliographie

[Abdallah et Lesser, 2005] Abdallah, S. et Lesser, V. (2005). Modeling Task Allocation Using a Decision Theoretic Model. *In Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 719–726, Utrecht, Netherlands. ACM Press.

[Ai-Chang *et al.*, 2004] Ai-Chang, M., Bresina, J., Charest, L., Chase, A., Cheng-jung Hsu, J., Jonsson, A., Kanefsky, B., Morris, P., Rajan, K., Yglesias, J., Chafin, B. G., Dias, W. C. et Maldague, P. F. (2004). Mapgen : Mixed-initiative planning and scheduling for the mars exploration rover mission. *In IEEE Intelligent Systems*, volume 19, pages 8–12.

[Asimov, 2004] Asimov, I. (2004). *Le cycle des robots*. J'ai Lu, première parution 1940.

[Astrom, 1965] Astrom, K.J.. (1965). Optimal control of markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:403–406.

[Austin, 1962] Austin, J.L. (1962). *How to do Things with Words*. Clarendon Press, traduction française Quand dire c'est faire, Le Seuil, 1970.

[Becker *et al.*, 2004a] Becker, R., Lesser, V. et Zilberstein, S. (2004a). Decentralized Markov Decision Processes with Event-Driven Interactions. *In The Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, volume 1, pages 302–309, NYC. IEEE Computer Society.

[Becker *et al.*, 2005] Becker, R., Lesser, V. et Zilberstein, S. (2005). Analyzing Myopic Approaches for Multi-Agent Communication. *In Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 05)*, pages 550–557, Compiegne, France. IEEE Computer Society.

[Becker *et al.*, 2003] Becker, R., Zilberstein, S., Lesser, V. et Goldman, C. (2003). Transition-independent decentralized markov decision processes. *In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 41–48, Melbourne, Australia. ACM Press.

[Becker *et al.*, 2004b] Becker, R., Zilberstein, S., Lesser, V. et Goldman, C. (2004b). Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

[Bernstein *et al.*, 2005] Bernstein, D., Hansen, E.A. et Zilberstein, S. (2005). Bounded policy iteration for decentralized pomdps. *In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland.

[Bernstein *et al.*, 2002] Bernstein, D., Zilberstein, S. et Immerman, N. (2002). The complexity of decentralized control of mdps. *In Mathematics of Operations Research*, pages 27(4) :819–840.

[Bernstein *et al.*, 2001] Bernstein, D., Zilberstein, S., Washington, R. et Bresina, J. (2001). Planetary rover control as a markov decision process. *In The 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Montreal, Canada.

[Beynier, 2003] Beynier, A. (2003). Rapport préliminaire : Mdps interactifs et contraintes temporelles. *In Proceedings des troisièmes journées nationales PDMIA'2003*, Caen.

[Beynier *et al.*, 2006] Beynier, A., Jeanpierre, L. et Mouaddib, A.I. (2006). Optimal planning for autonomous agents under time and resource uncertainty. *In Third International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Portugal.

[Beynier et Mouaddib, 2004a] Beynier, A. et Mouaddib, A.I. (2004a). Decentralized markov decision processes for handling temporal and resource constraints in a multiple robot system. *In Proceedings of the 7th International Symposium on Distributed Autonomous Robotic System (DARS)*.

[Beynier et Mouaddib, 2004b] Beynier, A. et Mouaddib, A.I. (2004b). A decentralized multiagent decision approach for handling temporal and resource constraints : preliminary report. *In Proceedings of AAAI Symposium on Bridging the Multi-Agent and Multi-Robotic Research Gap*.

[Beynier et Mouaddib, 2004c] Beynier, A. et Mouaddib, A.I. (2004c). Non-communicative dec-mdp for cooperative multi-agent systems. *In ECAI Workshop on Multi-Agent Decision Processes : Theories and Models*, pages 8–15.

[Beynier et Mouaddib, 2004d] Beynier, A. et Mouaddib, A.I. (2004d). Processus décisionnels de markov décentralisés et interactifs avec contraintes temporelles. *Journal Eléctronique d'Intelligece Artificielle (JEDAI)*.

[Beynier et Mouaddib, 2005a] Beynier, A. et Mouaddib, A.I. (2005a). Un algorithme polynomial pour la résolution de mdp décentralisés. *In Modèles Formels de l'Interaction (MFI'05)*, pages 35–44.

[Beynier et Mouaddib, 2005b] Beynier, A. et Mouaddib, A.I. (2005b). A polynomial algorithm for decentralized markov decision processes with temporal constraints. *Proceedings of*

the fourth Interantional Joint Conference on Autonomous Agents and MultiAgent Systems, pages 963–969.

[Beynier et Mouaddib, 2006a] Beynier, A. et Mouaddib, A.I. (2006a). An iterative algorithm for solving constrained decentralized markov decision processes. *In The Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

[Beynier et Mouaddib, 2006b] Beynier, A. et Mouaddib, A.I. (2006b). An iterative algorithm for solving constrained decentralized markov decision processes. *In AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*.

[Blythe, 1999a] Blythe, J. (1999a). Decision-theoretic planning. *AI Magazine*.

[Blythe, 1999b] Blythe, J. (1999b). *Planning under uncertainty in Dynamic domains*. Phd thesis, Carnegie Mellon University.

[Bonabeau *et al.*, 1999] Bonabeau, E., Dorigo, M. et G., T. (1999). *Swarm Intelligence : From Natural to Artificial Systems*. Oxford University Press.

[Boutilier, 1996] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. *In Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'96)*, pages 195–201.

[Boutilier *et al.*, 1999] Boutilier, C., Dean, T. et Hanks, S. (1999). Decision-theoretic planning : Structural asumptions and computational leverage. *Journal of Articicial Intelligence Research*, 1:1–93.

[Boutilier et Dearden, 1994] Boutilier, C. et Dearden, R. (1994). Using abstractions for decision-theoretic planning with time constraints. *In AAAI'94 : Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, pages 1016–1022, Menlo Park, CA, USA. American Association for Artificial Intelligence.

[Boutilier *et al.*, 1995] Boutilier, C., Dearden, R. et Goldszmidt, M. (1995). Exploiting structure in policy construction. *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111.

[Boutilier, 1999] Boutilier, G. (1999). Sequential optimality and coordination in multiagent systems. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 478–485.

[Bresina *et al.*, 2002] Bresina, J., Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. et Washington, R. (2002). Planning under continuous time and resource uncertainty : A challenge for ai. *In UAI*.

[Bresina et Washington, 2000] Bresina, J. et Washington, R. (2000). Expected utility distributions for flexible contingent execution. *In AAAI Workshop on Representation issues for Real World Planning system*.

[Capek, 1997] Capek, K. (1997). *Rossum Universal Robots*. L'aube, première parution 1921.

[Cardon *et al.*, 2001] Cardon, S., Mouaddib, A., Zilberstein, S. et Washington, R. (2001). Adaptive control of acyclic progressive processing task structures. *In Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI-2001*, pages 701–706.

[Chadès *et al.*, 2002] Chadès, I., Scherrer, B. et Charpillet, F. (2002). A heuristic approach for solving decentralized-POMDP : Assessment on the pursuit problem. *In Proceedings of the Sixteenth ACM Symposium on Applied Computing*.

[Damiani, 2005] Damiani, S. (2005). *Gestion d'une constellation de staellites de surveillance de la terre : autonomie et coordination*. Phd thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace.

[Damiani *et al.*,] Damiani, S., Verfaillie, G. et Charmeau, M. An earth watching satellite constellation : how to manage a team of watching agents with limited communications. *In Proceedings of the fourth Interantional Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2005)*, pages 455–462.

[Dean *et al.*, 1993] Dean, T., Kaelbling, L. P., Kirman, J. et Nicholson, A. (1993). Planning with deadlines in stochastic domains. *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 574–579, Menlo Park, California. AAAI Press.

[Dean et Lin, 1995] Dean, T. et Lin, S. (1995). Decomposition techniques for planning in stochastic domains. *In IJCAI-95*.

[Dearden et Boutilier, 1994] Dearden, R. et Boutilier, C. (1994). Integrating planning and execution in stochastic domains. *In In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 162–169.

[Decker et Lesser, 1993a] Decker, K. et Lesser, V. (1993a). Quanititative modeling of complex computational task environnement. *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224.

[Decker et Lesser, 1993b] Decker, K. et Lesser, V. (1993b). Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting Finance and Management*, 2(4):215–234.

[Demazeau, 1997] Demazeau, Y. (1997). Steps towards multi-agent oriented programming. *In First International Workshop on Multi-Agent Systems (IWMAS'97)*, Boston.

[Drake, 1962] Drake, r. (1962). *Observation of a Markov process through a noisy channel*. Phd thesis, Massachusetts Institute of Technology.

[Dutech *et al.*, 2001] Dutech, A., Buffet, O. et Charpillet, F. (2001). Multi-agent systems by incremental gradient reinforcement learning. *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 833–838.

[Emery-Montemerlo *et al.*, 2004] Emery-Montemerlo, R., Gordon, G., Schneider, J. et Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. *In Proceedings of the Third Joint Conference on Autnomous Agents and Multi Agent Systems*.

[Esben *et al.*, 2002] Esben, H. O., Maja, J. M. et Gaurav, S. S. (2002). Multi-robot task allocation in the light of uncertainty. *In Proceedongs of IEEE International Conference on Robotics and Automation ICRA 2002*, pages 3002–3007.

[Estlin *et al.*, 1999] Estlin, T., Tobias, A., Rabideau, G., Castana, R., Chien, S. et Mjolsness, E. (1999). An integrated system for multi-rover scientific exploration. *In The Sixteenth National Conference on Artificial Intelligence*.

[Ferber, 1995] Ferber, J. (1995). *Les Systèmes multi-agents : Vers une intelligence collective*. Inter Editions.

[Ferber et Muller, 1995] Ferber, J. et Muller, J.P. (1995). Influences and reaction : A model of situated multiagent systems. *In Proceedings of the International Conference on Multi-Agent Systems (ICMAS)*.

[Finin *et al.*, 1994] Finin, T., Fritzson, R., McKay, D. et McEntire, R. (1994). Kqml as an agent communication language. *In Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA. ACM Press.

[FIPA, 2006] FIPA (2006). http ://www.fipa.org.

[Fox, 1981] Fox, M.S. (1981). An organizational view of distributed systems. *In IEEE Transactions on Systems, Man, and Cybernetics*, volume 11, pages 70–80.

[Franklin et Graesser, 1996] Franklin, S. et Graesser, A. (1996). Is it an agent, or just a program ? : A taxonomy for autonomous agents. *In Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35, London, UK. Springer-Verlag.

[Gerkey et Matarić, 2002] Gerkey, B. P. et Matarić, M. J. (2002). Sold ! : Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768.

[Goldman et Zilberstein, 2003] Goldman, C. et Zilberstein, S. (2003). Optimizing information exchange in cooperative multiagent systems. *In International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 137–144.

[Goldman et Zilberstein, 2004] Goldman, C. et Zilberstein, S. (2004). Decentralized control of cooperative systems : Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174.

[Gordon, 1999] Gordon, G. (1999). *Approximate Solutions to Markov Decision Processes.* Thèse de doctorat, Carnegie Mellon University.

[Hanna, 2003] Hanna, H. (2003). *La planification des tâches et des ressources sous incertitude dans un système multi-agent.* Phd thesis, Université de Caen.

[Hanna et Mouaddib, 2002] Hanna, H. et Mouaddib, A. (2002). Task selection as decision making in multiagent system. *In AAMAS*, pages 616–623.

[Hansen *et al.*, 2004] Hansen, E.A., Bernstein, D. et Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. *In Proceedings of the Nineteenth National Conference on Artificial Intelligence.*

[Hauskrecht *et al.*, 1998] Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T. et Boutilier, C. (1998). Hierarchical solution of Markov decision processes using macro-actions. *In Uncertainty in Artificial Intelligence*, pages 220–229.

[Hewitt, 1986] Hewitt, C. (1986). Offices are open systems. *ACM Trans. Inf. Syst.*, 4(3):271–287.

[Howard, 1960] Howard, R. A. (1960). *Dynamic Programming and Markov Processes.* MIT Press.

[Hu et Wellman, 1998a] Hu, J. et Wellman, M. P.. (1998a). Multiagent reinforcement learning : Theoretical framework and an algorithm. *In ICML '98 : Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Hu et Wellman, 1998b] Hu, J. et Wellman, M. P.. (1998b). Online learning about other agents in a dynamic multiagent system. *In AGENTS '98 : Proceedings of the second international conference on Autonomous agents*, pages 239–246, New York, NY, USA. ACM Press.

[Jennings *et al.*, 1998] Jennings, N. R., Sycara, K. et Wooldridge, M. (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38.

[K-Team, 2001] K-Team (2001). *Koala User Manual.*

[Kaelbling *et al.*, 1998] Kaelbling, L. P., Littman, M. L. et Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

[Kaelbling *et al.*, 1996] Kaelbling, L. P., Littman, M. L. et Moore, A. P. (1996). Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Koller et Milch, 2003] Koller, D. et Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, pages 45(1) : 181–221.

[Laroche, 2000] Laroche, P. (2000). *Processus Décisionnels de Markov appliqués à la planification sous incertitude.* Thèse de doctorat, Université Henri Poincaré Nancy 1.

[Levin *et al.*, 1998] Levin, E., Pieraccini, R. et Eckert, W. (1998). Using markov decision process for learning dialogue strategies. *In Proceedings of the IEEE, Transactions on Speech and Audio Processing*, volume 8, pages 11–23.

[Littman *et al.*, 1995] Littman, M. L., Dean, T. L. et Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. *In Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, pages 394–402, Montreal, Québec, Canada.

[Littman, 1994] Littman, M. L.. (1994). Markov games as a framework for multi-agent reinforcement learning. *In Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ. Morgan Kaufmann.

[Littman, 1996] Littman, r. (1996). *Algorithms for Sequential Decision Making.* Phd thesis, Brown University.

[Malone, 1988] Malone, T.W. (1988). What is coordination theory. *In National Science Foundation Coordination Theory Workshop.*

[Mishkin *et al.*, 1998] Mishkin, A., Morrison, J., Nguyen, T., Stone, H., Cooper, B. et Wilcox, B. (1998). Experiences with operations and autonomy of the mars path nder microrover. *In Proceedings of IEEE Aerospace Conference*, volume 2, pages 337–351.

[Morimoto, 2000] Morimoto, T. (2000). *How to develop a RoboCupRescue agent.* RoboCupRescue Technical Committee.

[Mouaddib et Zilberstein, 1998] Mouaddib, A.-I. et Zilberstein, S. (1998). Optimal scheduling for dynamic progressive processing. *In ECAI-98*, pages 499–503.

[Mundhenk *et al.*, 2000] Mundhenk, M., Goldsmith, J., Lusena, C. et Allender, E. (2000). Complexity of finite-horizon markov decision process problems. *J. ACM*, 47(4):681–720.

[Murphy, 2000] Murphy, K. (2000). A survey of POMDP solution techniques. *Technical Report, U.C. Berkeley.*

[Nair *et al.*, 2005] Nair, R., Pradeep, V., Milind, T. et Makoto, Y. (2005). Networked distributed POMDPs : A synthesis of distributed constraint optimization and POMDPs. *In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05).*

[Nair *et al.*, 2004] Nair, R., Roth, M., Yokoo, M. et Tambe, M. (2004). Communication for improving policy computation in distributed pomdps. *In Proceedings of the Third International Joint Conference on Agents and Multiagent Systems (AAMAS-04)*, pages 1098–1105.

[Nair *et al.*, 2003] Nair, R., Tambe, M., Yokoo, M., Marsella, S. et Pynadath, D.V. (2003). Taming decentralized pomdps : Towards efficient policy computation for multiagent settings. *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 705–711.

[Nash, 1950] Nash, J. (1950). Equilibrium points in n-person games. *In Proceedings of the National Academy of the USA*, volume 36(1), pages 48–49.

[Papadimitriou et Tsitsiklis, 1987] Papadimitriou, C.H. et Tsitsiklis, J.N. (1987). The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450.

[Pearl, 1981] Pearl, J. (1981). Probabilistics reasoning in intelligent systems : Networks of plausible influence. *Morgan-kaufmann.*

[Peshkin *et al.*, 2000] Peshkin, L., Kim, K., Meuleu, N. et Kaelbling, L. (2000). Learning to cooperate via policy search. *In Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–314.

[Puterman, 2005] Puterman, M. L. (2005). *Markov Decision processes : discrete stochastic dynamic programming.* Wiley-Interscience, New York, deuxième édition.

[Pynadath et Tambe, 2002] Pynadath, D. et Tambe, M. (2002). The communicative multiagent team decision problem : Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, pages 389–423.

[Rabiner et Juang, 1993] Rabiner, L. et Juang, B.-H. (1993). *Fundamentals of speech recognition.* Prentice Hall Signal Processing Series, Upper Saddle River, NJ, USA.

[Roy *et al.*, 2000] Roy, N., Pineau, J. et Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong.

[Russell et Norvig, 2003] Russell, S. et Norvig, P. (2003). *Artificial Intelligence : A Modern Approach.* Prentice Hall Series.

[Sandholm, 1998] Sandholm, T. (1998). Contract types for satisficing task allocation : I theoretical results. *In Proceedings of AAAI 1998 Spring Symposium : Satisficing Models.*

[Sandholm et Lesser, 1995] Sandholm, T. et Lesser, V. (1995). Issues in automated negotiation and electronic commerce : Extending the contract net framework. *In First ICMAS*, pages 328–335.

[Searle, 1969] Searle, J.R. (1969). *Speech Acts.* Cambridge University Press Clarendon Press, traduction française Les actes de langage, 1972, Hermann, coll. Savoir : lettres.

[Shannon et Weaver, 1948] Shannon, C. et Weaver, W. (1948). *The Mathematical theory of Communication.* University of Illinois Press.

[Shapley, 1953] Shapley, L.S.. (1953). Stochastic games. *National Academy of Sciences of the United States of America*, 39:1095–1100.

[Shelley, 2005] Shelley, M. (2005). *Frankenstein ou le Prométhée moderne.* J'ai Lu, première parution 1818.

[Shoham et Tennenholtz, 1992] Shoham, Y. et Tennenholtz, M. (1992). On the synthesis of useful social laws for artificial agent societies. *In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 276–281, San Jose, California.

[Simmons et Koenig, 1995] Simmons, R. et Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087.

[Simon, 1947] Simon, H. (1947). *Adminsitrative behavior. A study of Decision-Making Processes in Administrative Organisation.* Free Press.

[Smallwood et Sondik, 1973] Smallwood, R.D.. et Sondik, E.J.. (1973). The optimal control of partially observable markov decicion processes over a finite horizon. *Operation Research*, 21:1071–1088.

[Smith, 1980] Smith, R.G. (1980). The contract net protocol : High-level communication and control in a distributed problem solver. *In IEEE Transactions on Computers*, volume 29, pages 1104–1113.

[Sutton et Barto, 1998] Sutton, R. et Barto, A. (1998). Reinforcement learning : An introduction. *MIT press, Cambrige, MA.*

[Sutton *et al.*, 1999] Sutton, R., Precup, D. et Singh, S. (1999). Between MDPs and semi-MDPs : A framework of temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

[Szer *et al.*, 2005] Szer, D., Charpillet, F. et Zilbersetin, S. (2005). MAA* : A heuristic search algorithm for solving decentralized POMDPs. *In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'2005).*

[The RoboCup Rescue Technical Committee, 2000] The RoboCup Rescue Technical Committee (2000). *RoboCup-Rescue simulator manual.*

[Thrun *et al.*, 1998] Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W. et Kuipers, B. (1998). Integrating topological and metric maps for mobile robot navigation : A statistical approach. *In Proceedings of the National Conference on Artificial Intelligence (AAAI).*

[Turner, 2005] Turner, R. M. (2005). Intelligent mission planning and control of autonomous underwater vehicles. *In Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems (ICAPS 05)*, pages 5–8, Monterey, California.

[Von Neumann et Morgenstern, 1944] Von Neumann, J. et Morgenstern, O. (1944). *Theory of games and economic behavior.* Princeton University Press.

[Weiss, 1999] Weiss, G. (1999). *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence.* MIT Press.

[Wieser, 1889] Wieser, F. (1889). *Valeur naturelle (Der natürliche Wert).*

[Wooldridge, 2002] WOOLDRIDGE, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley and Sons.

[Wooldridge et Jennings, 1995] WOOLDRIDGE, M. et JENNINGS, N. R. (1995). Intelligent agents : Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.

[Xuan et Lesser, 2002] XUAN, P. et LESSER, V. (2002). Multi-agent policies : From centralized ones to decentralized ones. *In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1098–1105. ACM Press.

[Xuan *et al.*, 2001] XUAN, P., LESSER, V. et ZILBERSTEIN, S. (2001). Communication decisions in multiagent cooperation : Model and experiments. *In Proceedings of the Fifth International Conference on Autonomous Agents*, pages 616–623, Montreal. ACM Press.

[Young, 1990] YOUNG, S.R. (1990). Use of dialogue, pragmatics and semantics to enhance speech recognition. *Speech Commun.*, 9(5-6):551–564.

[Zilberstein et Mouaddib, 2000] ZILBERSTEIN, S. et MOUADDIB, A.-I. (2000). Optimizing resource utilization in planetary rovers. *In 2nd NASA Workshop on Planning and Scheduling for Space*, pages 163–168.

[Zilberstein *et al.*, 2000] ZILBERSTEIN, S., MOUADDIB, A.-I. et ARNT, A. (2000). Dynamic scheduling for progressive processing. *In Workshop ECAI-00*, pages 139–145.

[Zilberstein et Mouaddib, 1999] ZILBERSTEIN, S. et MOUADDIB, A.I. (1999). Reactive control for dynamic progressive processing. *In IJCAI-99*, pages 1269–1273.

[Zilberstein *et al.*, 2002] ZILBERSTEIN, S., WASHINGTON, R., BERNSTEIN, D. et MOUADDIB, A. (2002). *Decision-Theoretic Control of Planetary Rovers*, volume 2466, pages 270–289. M. Beetz et al. (Eds.) : Plan-Based control of Robotic Agents, LNAI.

Résumé

Cette thèse porte sur la prise de décision distribuée dans des systèmes multi-agents agissant sous incertitude (les colonies de robots autonomes par exemple). Les processus décisionnels de Markov Décentralisés décrivent un formalisme mathématique permettant de modéliser et de résoudre de tels problèmes. Leur utilisation pour la planification des tâches dans des applications réelles pose toutefois quelques difficultés. Le modèle usuel des DEC-MDPs ne permet par exemple pas la prise en compte de contraintes sur l'exécution des tâches. De plus, la complexité de leur résolution est telle qu'il est difficile de déterminer une solution optimale excepté pour de petits problèmes.

Le travail que nous présentons dans cette thèse a pour premier objectif d'adapter le modèle des DEC-MDPs afin de proposer une modélisation adéquate du temps et des actions, et de permettre la représentation de problèmes réels. Nous décrivons ainsi une nouvelle classe de DEC-MDPs : les OC-DEC-MDPs (DEC-MDP avec Coût Occasionné). Dans un second temps, nous nous intéressons à leur résolution. Nous proposons différents algorithmes procédant à la planification des tâches de chaque agent en vue d'une prise de décision décentralisée et autonome, en accord avec les contraintes du problème. Afin de développer des algorithmes efficaces et de traiter des problèmes de taille importante, nous recherchons une approximation de la solution optimale. Nous procédons également à un découpage du problème initial en un ensemble de MDPs, et introduisons la notion de coût occasionné afin de tenir compte des interactions entre les agents et de calculer des politiques coopératives.

Mots-clés: Processus Décisionnels de Markov, Systèmes multi-agents, Planification sous incertitude, Intelligence Artificielle, Robotique collective

Abstract

Title : An approach to solve Decentralized Markov Decision Processes with temporal constraints

This thesis deals with distributed multiagent decision-making under uncertainty. We formalize this problem with Decentralized Markov Decision Processes (DEC-MDP) which extends Markov Decision Processes (MDP) to multi-agent settings. Even if DEC-MDPs describe an expressive framework for cooperative multiagent decision, they suffer from a high complexity and fail to formalize constraints on task execution. Despite the wide variety of approaches to solve DEC-MDPs, computing a solution for large problems remains a serious challenge even for approximation approaches.

We develop an approach that can solve large problems, and that can deal with more complex time and action representations. We therefore define a class of DEC-MDP, OC-DEC-MDP, that allows us to consider several possible durations for each task taking into account constraints on task execution. Having considered the representation of the problems we deal with, we turn to OC-DEC-MDP resolution. Our purpose is to develop an efficient planning approach that computes each agent's policy even for large missions. Given the high complexity of finding an optimal solution, we aim at computing an approximate solution. We also split the multiagent decision problem into a set of MDPs. For purposes of coordinating the agents, we then introduce the notion of Opportunity Cost.

Keywords: Markov Decision Processes, Multiagent Systems, Planning under uncertainty, Artificial Intelligence, Robotics

GREYC - CNRS UMR 6072
Université de Caen, Campus II
BP 5186, F-14032 Caen cedex, France