Modèle décisionnel orienté comportement fondé sur le vote : Application à la navigation d'agents autonomes en environnement simulé

David Hanon

LAMIH
LABORATOIRE D'AUTOMATIQUE DE MECANIQUE ET D'INFORMATIQUE INDUSTRIELLES ET HUMAINES

Numéro d'ordre : 06 / 18

THÈSE

Présentée à

L'Université de Valenciennes et du Hainaut-Cambrésis

en vue de l'obtention du grade de

DOCTEUR

Mention Automatique et Informatique des Systèmes Industriels et Humains

Spécialité Informatique

Par

David HANON

Modèle décisionnel orienté comportement fondé sur le vote : Application à la navigation d'agents autonomes en environnement simulé.

Soutenue publiquement le 12 décembre 2006 devant le Jury composé de :

M. Rachid Alami, Directeur de recherches, CNRS, Toulouse	Rapporteur
M. Stéphane Espié, Chargé de recherches (HDR), INRETS, Arcueil	Examinateur
Mme Emmanuelle Grislin-le Strugeon, Maître de Conférences, Université de Valenciennes	Co-Directeur
Mme Zahia Guessoum, Maître de Conférences (HDR), Université de Reims	Rapporteur
M. René Mandiau, Professeur, Université de Valenciennes	Directeur
M. Sylvain Piechowiak, Professeur, Université de Valenciennes	Président
M. Jean-Christophe Routier, Maître de Conférences (HDR), Université de Lille I	Examinateur

LAMIH
LABORATOIRE D'AUTOMATIQUE DE MECANIQUE ET D'INFORMATIQUE INDUSTRIELLES ET HUMAINES

Numéro d'ordre : 06 / 18

THÈSE

Présentée à

L'Université de Valenciennes et du Hainaut-Cambrésis

en vue de l'obtention du grade de

DOCTEUR

Mention Automatique et Informatique des Systèmes Industriels et Humains

Spécialité Informatique

Par

David HANON

Modèle décisionnel orienté comportement fondé sur le vote : Application à la navigation d'agents autonomes en environnement simulé.

Soutenue publiquement le 12 décembre 2006 devant le Jury composé de :

M. Rachid Alami, Directeur de recherches, CNRS, Toulouse	Rapporteur
M. Stéphane Espié, Chargé de recherches (HDR), INRETS, Arcueil	Examinateur
Mme Emmanuelle Grislin-le Strugeon, Maître de Conférences, Université de Valenciennes	Co-Directeur
Mme Zahia Guessoum, Maître de Conférences (HDR), Université de Reims	Rapporteur
M. René Mandiau, Professeur, Université de Valenciennes	Directeur
M. Sylvain Piechowiak, Professeur, Université de Valenciennes	Président
M. Jean-Christophe Routier, Maître de Conférences (HDR), Université de Lille I	Examinateur

Remerciements

Le travail présenté dans ce rapport a été réalisé au Laboratoire d'Automatique, de Mécanique, et d'Informatique industrielles et Humaines et plus particulièrement au sein de l'équipe Raisonnement Automatique et Interaction Homme-Machine, dirigée par le professeur Kolski. Je tiens à remercier les personnes qui ont contribué à son aboutissement.

Je remercie mes directeurs de thèse pour leur disponibilité et leur soutien. Je remercie René Mandiau de m'avoir fait partager son expérience en matière de recherche et sa connaissance des agents. Je souhaite exprimer ma gratitude à Emmanuelle Grislin pour m'avoir encadré, m'avoir aidé à développer puis clarifier les quelques idées qui font l'objet de ce rapport.

Je tiens à remercier madame Zahia Guessoum, maître de conférences à l'université de Reims, ainsi que monsieur Rachid Alami, directeur de recherches au CNRS, LAAS (Laboratoire d'Analyse et d'Architecture des systèmes) de m'avoir fait l'honneur d'être les rapporteurs de ce mémoire.

Mes remerciements s'adressent également à messieurs Stéphane Espié, chargé de recherches à l'INRETS, Sylvain Piechowiak, professeur à l'université de Valenciennes et Jean-Christophe Routier, maître de conférences à l'université de Lille 1, qui ont accepté d'examiner ce travail.

Je remercie les personnes ayant participé à RESPECT : Madame Granié et Messieurs Charron, Dos Santos, Ebel, Jung, Lepoutre, Pudlo et Stanciulescu.

Je remercie, par l'intermédiaire du CNRS et de la région Nord – Pas de Calais, l'ensemble des contribuables français et plus particulièrement nordistes d'avoir financé cette thèse.

Je souhaite également remercier ma famille, et en particulier mon père pour sa présence, son soutien et son aide. Sa participation en tant que « directeur de recherche des fautes d'orthographe » m'a été extrêmement précieuse.

Je remercie mes amis, Abdelwaheb, Amine, Anli, Anne, Arnaud, Bruno, Elisa, Gilles, Hervé, Jay, Marie, Patoue, Philippe, Mathieu (et ceux que j'oublie à cet instant) pour m'avoir soutenu lorsque les piétons piétinaient. J'adresse mes sentiments les plus sincères à Benoît Paul ainsi qu'à Mélanie Joly pour leur sympathie et leurs arbres.

Enfin, je remercie Céline pour sa présence.

Table des matières

Introduction générale

Le travail présenté dans ce rapport s'inscrit dans le cadre de l'utilisation d'agents autonomes pour la simulation de piétons. Leur autonomie se traduit, en particulier, par leur capacité à prendre eux-mêmes des décisions. Or, ce problème est particulièrement ardu parce que continu (possibilités indénombrables) et dynamique (les données sur lesquelles se base le raisonnement évoluent). Dans ce rapport nous étudions ce problème dans le cadre des architectures orientées comportement et plus précisément celles coordonnées par le vote.

Cette introduction générale est dédiée à la présentation du contexte applicatif et les motivations de ce travail. Il s'agit de la réalisation d'un prototype de simulateur dédié à l'apprentissage de la sécurité routière chez les enfants. Notre contribution à ce projet concerne le comportement des piétons évoluant de manière autonome dans la simulation. Ces personnages apportent une réelle valeur ajoutée à la formation sur simulateur. Néanmoins, leur développement reste un problème complexe et ouvert en raison des nombreuses contraintes imposées par l'application.

Le projet RESPECT

L'insécurité routière est un problème récurrent en France. La répression fait (semble t-il) ses preuves, mais la prévention et la formation ne doivent pas être négligées. La formation des conducteurs est longue, coûteuse et fait appel à des professionnels. Qu'en est-il, aujourd'hui, de la formation des piétons et en particulier des enfants ?

Le projet PREDIT[1] « Route Empruntée en Sécurité par le Piéton-Enfant Confronté au Trafic » (RESPECT) est le point de départ de notre étude. Il constitue une proposition originale pour la formation des jeunes piétons. Les enfants sont, en effet, une population fortement touchée par les accidents de la route (cf. [Granié et al., 2001]), en particulier parce qu'ils sont peu visibles par les conducteurs, et qu'ils disposent d'une expérience limitée. Le projet RESPECT a duré 2 ans (d'août 2001 à août 2003) et a permis la réalisation d'un prototype de simulateur éducatif. Ce simulateur reproduit l'environnement urbain et peut être utilisé pour sensibiliser les enfants à la sécurité routière. Pour cela, il permet de placer

1 Programme de REcherche et D'Innovation dans les Transports terrestres. Le PREDIT est un programme de recherche, d'expérimentation et d'innovation dans les transports terrestres, initié et conduit par les ministères chargés de la recherche, des transports, de l'environnement et de l'industrie, l'ADEME (Agence De l'Environnement et de la Maîtrise de l'Energie) et l'ANVAR (Agence Nationale de Valorisation de la Recherche)." Le PREDIT 3 est encadré par trois objectifs généraux :
- assurer une mobilité durable des personnes et des biens,
- accroître la sécurité des systèmes de transport,
- réduire les impacts environnementaux et contribuer à la lutte contre l'effet de serre.

l'enfant-joueur dans des situations reconnues comme étant sources d'accidents. La figure 1 est une copie d'écran du simulateur RESPECT, décrivant des piétons se déplaçant dans un contexte urbain.

Figure 1 : *Le simulateur RESPECT.*

Cette idée d'utiliser l'informatique et la réalité virtuelle dans un but pédagogique n'est pas nouvelle et on nomme ces logiciels des « Environnements Interactifs d'Apprentissage par Ordinateur » (EIAO) ou « Environnements Interactifs pour l'Apprentissage Humain » (EIAH). Le principal apport des environnements virtuels est de permettre, aux personnes formées, d'agir sur un environnement simulé. Cela favorise l'apprentissage de savoir-faire et non plus simplement de connaissances théoriques. Si la réalité virtuelle apporte de nouvelles possibilités de formation, elle est également très attractive. Une étude [Fluch et al., 2003] montre même que les formations utilisant la réalité virtuelle donnent de meilleurs résultats car les apprenants sont plus motivés et y consacrent plus de temps. Nous n'entrerons pas davantage dans les détails concernant les EIAH. Une introduction complète du domaine est présentée dans le chapitre 7 de [Fluchs et al., 2003].

Le simulateur RESPECT, prototype d'un EIAH, a été développé comme un support pédagogique mis à la disposition du formateur. Outre les outils d'édition de scénario et réseau routier, le simulateur offre deux modes de fonctionnement : un mode de simulation (jeu) et un mode « rejeu » qui permet de revoir l'action. Pendant la phase de jeu, l'élève agit dans la simulation au travers de son avatar (c'est à dire un personnage qui le représente). Il a alors une vue subjective de la scène. Pendant la phase de rejeu, il est possible de visualiser la scène depuis n'importe quel point de vue (y compris le point de vue d'un conducteur ou d'un autre piéton). Cela permet d'exploiter les erreurs de l'apprenant. Le formateur peut, par exemple, montrer à l'élève qu'il n'était pas bien positionné et donc qu'il n'était pas visible par certains conducteurs.

Les compétences scientifiques et techniques nécessaires à la réalisation d'un tel projet sont nombreuses. Notre équipe était chargée du pilotage des mobiles virtuels à base d'agents ; les autres partenaires qui sont intervenus dans la conception et la réalisation du logiciel réunissent l'ensemble des compétences requises pour la spécification et la réalisation du simulateur éducatif :

- Madame Granié, Laboratoire de Psychologie de la Conduite (LPC-INRETS) a apporté son savoir-faire en sécurité routière et en psychologie du développement des compétences routières. Elle a notamment mené les études conduisant à la spécification du logiciel et des scénari simulés.
- Monsieur Charron, du Centre de Recherches en Psychologie, Cognition et Communication (CRP2C) de l'université de Rennes 2 a apporté ses compétences en matière de pédagogie et psychologie. En particulier, il a spécifié et conçu les scénarii.
- Monsieur Jung, chef de projet et son équipe de l'entreprise CORYS T.E.S.S. (Training & Engineering Support Systems), pilote du projet, ont apporté leur savoir-faire technologique dans la création de simulateurs de trafic et de conduite. Ils ont conçu et codé l'essentiel des modules du logiciel.
- Messieurs Lepoutre, Pudlo, Stanciulescu et Ebel, membres de l'équipe Biomécanique du LAMIH (dirigée par le professeur Lepoutre) ont apporté leurs compétences en biomécanique pour la simulation de la marche. Ils ont spécifié et conçu le module d'animation des piétons virtuels. Messieurs Stanciulescu et Ebel ont, en outre, codé ce module et réalisé les captures de mouvements à la base des animations.

Notre travail dans ce groupe concerne la simulation du comportement des piétons présents dans l'environnement simulé. Nous présenterons, dans ce cadre, son utilité et les difficultés liées à sa réalisation dans le paragraphe suivant.

Les piétons présents dans RESPECT

Notre contribution au projet RESPECT concerne le comportement des acteurs virtuels et, en particulier, les piétons présents dans la simulation. Ceux-ci jouent un rôle prépondérant dans le projet. Néanmoins leur conception s'avère délicate.

Les acteurs virtuels possèdent différents intérêts pédagogiques. En premier lieu, par leur simple présence, les acteurs virtuels contribuent au réalisme de l'application et donc au sentiment d'immersion. Une ville quasiment déserte ne serait pas réaliste. Or, il faut que le travail de transfert entre la simulation et la situation réelle soit minime afin que l'apprenant puisse réutiliser les compétences acquises. De plus, les acteurs virtuels participent pleinement aux scènes simulées car ils interagissent de manière directe avec l'apprenant. Par exemple, la présence de nombreux piétons sur le trottoir peut obliger l'élève à marcher sur la route. Enfin, l'utilité pédagogique des acteurs virtuels peut également être plus subtile. Ils peuvent être utilisés comme référence ou au contraire inciter à la faute. [Fluch et al., 2003] cite Frasson qui « a exploré l'utilisation d'un fauteur de troubles automatisé [...] qui fournit parfois des informations incorrectes afin de tester la confiance en soi de l'apprenant ». Pour notre application de sécurité routière, il est opportun de disposer de piétons aux comportements différenciés. Cela permet, par exemple, à un formateur de vérifier si un élève est capable de ne pas suivre un piéton virtuel imprudent.

Comme l'énonçait encore récemment D. Thalmann [Thalmann, 2003], il reste difficile de gérer le comportement des piétons virtuels, en particulier lorsqu'on veut les rendre autonomes. D'ailleurs, outre son équipe du VRLab[2] (anciennement MIRALab) de l'université de Genève, de nombreux groupes s'attachent à ce problème : l'Equipe SIAMES[3] de l'IRISA de Rennes, par exemple. Afin de permettre l'immersion du joueur dans la simulation, les piétons virtuels doivent être capables de se déplacer dans les scènes urbaines en montrant un comportement le plus réaliste possible. Pour cela il leur faut s'adapter à leur environnement selon deux aspects supplémentaires : la configuration de la ville (disposition des rues, des passages piétons, …) et la situation présente (arrivée d'un véhicule, panne d'un feu tricolore, etc.).

La configuration de la ville n'est pas fixée *a priori*. Il est possible de créer de nouvelles villes correspondant à des besoins spécifiques. Il est donc important que les piétons disposent d'un raisonnement indépendant de la configuration de la ville afin qu'ils puissent construire un itinéraire et choisir les lieux de traversée.

Par ailleurs, la ville évolue durant la simulation : l'apprenant, les véhicules et les autres piétons se déplacent, les feux changent de couleur, etc. Les piétons doivent pouvoir réagir rapidement face à ces changements. Le temps de raisonnement accordé à chaque agent étant très court (de l'ordre de 10 millisecondes[4]), les algorithmes mis en oeuvre pour produire le raisonnement doivent être très efficaces. De plus, l'adaptation au joueur comporte une composante supplémentaire lorsqu'il est demandé de mettre celui-ci dans une situation donnée afin de réaliser un scénario déterminé.

Disposer de piétons réalistes, s'adaptant à la ville et aux situations, est un objectif ambitieux. En outre, il est intéressant d'offrir aux piétons une diversité de comportements en les rendant configurables sous différents aspects. Par exemple : un piéton prudent utilise le passage protégé et les feux, attend que les voitures s'arrêtent, etc ; un piéton imprudent traverse en courant, en dehors des passages cloutés et en obligeant les voitures à s'arrêter devant lui. Entre ces deux extrêmes, il est possible de définir une multitude de comportements piétonniers.

La coexistence d'un utilisateur imprévisible, de véhicules et de piétons autonomes, au sein d'un environnement urbain variable (selon le scénario choisi) et en perpétuelle évolution (changement d'états des feux) conduisent à envisager des modèles à base d'agents. En effet, les recherches sur les systèmes multi-agents ont, pour objet l'étude des systèmes au sein desquels différentes entités informatiques autonomes coexistent dans un même environnement.

En résumé, notre contexte d'étude se caractérise par une distribution d'entités autonomes en environnement dynamique. Or, il s'agit de caractéristiques fondamentales dans le cadre des recherches menées dans le domaine des systèmes multi-agents. C'est la raison qui a guidé le choix de l'approche multi-agents pour nos travaux, en vue d'une application au pilotage d'acteurs virtuels.

2 Virtual Reality Laboratory

3 Synthèse d'Images, Animation, Modélisation et Simulation, IRISA (Institut de Recherche en Informatique des Systèmes Aléatoires)

4 Pour produire une animation il faut au minimum 25 images par seconde d'où un temps de cycle inférieur à 40ms.

Plan du mémoire

Le premier chapitre a pour objectif de présenter l'état de l'art des agents et des acteurs virtuels. Comme nous venons de le montrer, les contraintes d'ouverture, les besoins en autonomie, la notion d'environnement virtuel nous amènent vers les systèmes multi-agents (SMA). Après avoir rappelé les principales définitions, nous brosserons rapidement les différentes facettes des SMA (Agent, Environnement, Interaction, Organisation et Utilisateur). L'animation comportementale est un domaine de recherche qui inclut la création d'acteurs virtuels. Les concepts introduits dans le cadre générique des SMA nous permettront de présenter ce domaine comme une application particulière. Afin d'être autonome, un personnage doit pouvoir prendre des décisions, autrement dit sélectionner des actions. Nous avons choisi de nous concentrer sur ce problème complexe et ouvert.

Le second chapitre aborde donc la question de la sélection d'actions. Après un rapide aperçu du problème, nous concentrerons notre étude sur les architectures orientées comportement. En effet, ces architectures sont dédiées aux environnements continus et dynamiques. Elles sont donc particulièrement intéressantes dans le cadre des acteurs virtuels. Les architectures orientées comportement se divisent en deux principales familles : arbitrage et fusion (à laquelle appartient le vote de préférences distribuées). Dans la dernière partie de ce chapitre nous établirons une synthèse des idées présentées. En particulier, nous montrerons les avantages du vote de préférences distribuées par rapport aux autres architectures. Cependant, les architectures actuelles utilisant le vote souffrent selon nous de plusieurs défauts.

Le troisième chapitre est consacré à la présentation de notre modèle décisionnel fondé sur les architectures orientées comportement coordonnées par le vote. Après avoir présenté les pistes d'amélioration que nous avons suivies, nous détaillerons l'architecture proposée. Nous décrirons donc les comportements ainsi que les données qu'ils manipulent (options candidates au vote et veto). Le fonctionnement de l'architecture est ensuite précisé. Nous avons scindé le cycle du modèle décisionnel en six étapes réparties entre les comportements et le mécanisme central de coordination.

Le quatrième chapitre concerne l'application et la validation du modèle dans le cadre de la navigation autonome (en environnement simulé). La description du problème de navigation nous amène à définir les tests réalisés ainsi que les critères mesurés. Le modèle général (présenté au troisième chapitre) est ensuite appliqué à la navigation. Lors de cette partie, nous présenterons l'implémentation du modèle ainsi que la conception des comportements spécifiques. Enfin, nous commenterons, les expérimentations réalisées en environnement statique puis dynamique.

Le dernier chapitre est consacré aux évolutions et extensions du travail présenté dans le rapport. Celles-ci concernent le modèle en lui-même ainsi que son application, dans le cadre de la navigation en environnement simulé ou dans d'autres domaines.

Chapitre I : État de l'art sur les acteurs virtuels

Ce premier chapitre a pour objectif de présenter les domaines des système multi-agents et de l'animation comportementale.

Les besoins et contraintes soulevés dans l'introduction générale, comme l'autonomie des piétons, leur cohabitation dans une même simulation, nous amènent aux concepts inhérents aux agents et systèmes multi-agents. Ceux-ci seront introduits dans la première section de ce chapitre. Après avoir rappelé les principales définitions, nous décrirons les différentes facettes des systèmes multi-agents (agent, environnement, interaction, etc.)

Nous nous appuierons sur les principes des systèmes multi-agents lors de la seconde section dédiée à la présentation de l'animation comportementale. Nous présenterons ce thème de recherche qui vise à augmenter la qualité des animations tout en réduisant l'intervention des animateurs. Cet objectif est réalisé par le développement d'entités autonomes nommées acteurs virtuels. Ces derniers sont, au même titre que les agents, dotés de mécanismes de raisonnement leur permettant, par exemple, de réagir aux actions des autres personnages ou de gérer leurs déplacements. Nous parcourrons donc le domaine en soulignant les problèmes résolus et ceux qui méritent d'être encore étudiés.

Nous conclurons ce chapitre en présentant le point que nous avons choisi de traiter, à savoir le modèle décisionnel. En effet, malgré les progrès réalisés pour augmenter la richesse des comportements des acteurs virtuels, nous pensons que ces modèles doivent être enrichis ; en particulier, pour prendre en compte l'aspect continu et dynamique des environnements virtuels. Les chapitres suivants seront donc plus particulièrement dédiés à ce problème.

1. Généralités sur les agents et systèmes multi-agents

Dans l'introduction générale, nous avons introduit notre application : des piétons virtuels et un apprenant évoluent dans des environnements virtuels et y interagissent. Cela nous conduit naturellement à l'étude des agents et systèmes multi-agents dont les grands principes sont abordés dans cette section.

1.1 Présentation des systèmes multi-agents

A l'origine des agents, il y a naturellement l'intelligence artificielle. Vers la fin des années 70, l'idée de faire résoudre des problèmes par, non plus un programme mais plusieurs, s'est développée. De cette volonté naquit l'intelligence artificielle distribuée (IAD). On établit alors cette nouvelle discipline en partant du principe que l'IAD s'attacherait aux principes abstraits (la coordination des solveurs) plutôt qu'aux détails de la résolution ou aux problèmes très concrets (tel que le parallélisme) (cf. chapitre 1 de [Briot et al., 2001]). L'expression « Système Multi-Agents » (SMA) s'est répandue comme un synonyme d'IAD. Comme nous le verrons par la suite, le champ d'application des SMA n'est (depuis longtemps) plus limité à la résolution distribuée de problèmes.

1.1.1. Agents

Beaucoup d'articles ont eu pour objectif de définir le terme « agent » et de présenter le domaine [Labidi et al., 1993]. Même si cela, nous semble aujourd'hui évident (et encore...), il convenait alors de délimiter un domaine de recherche naissant et d'éviter les usages abusifs ou erronés du terme. Carl Hewitt, cité dans [Wooldridge et al., 1995], remarquait, à l'époque, que cette question (« Qu'est ce qu'un agent ? ») n'était pas moins embarrassante que la question « Qu'est ce que l'intelligence ? ». Aujourd'hui, les usages abusifs sont courants et montrent, d'une certaine manière, l'expansion du domaine[5]. Nous retiendrons ici la définition formulée par [Ferber, 1995], qui décrit un agent comme une entité :

- capable de percevoir (de manière limitée) son environnement,
- ne disposant que d'une représentation partielle de son environnement,
- capable d'agir sur son environnement,
- pouvant communiquer avec d'autres agents,
- possédant des ressources propres,
- ayant des compétences et offrant éventuellement ses services,
- mue par un ensemble de tendances (objectifs personnels),
- cherchant à satisfaire ses objectifs en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception et de ses représentations.

Au même titre qu'on distingue l'IA forte et l'IA faible, Wooldridge propose deux définitions du terme agent [Wooldridge et al., 1995]. La définition « faible » fait référence à l'informatique. Un agent est une entité, le plus souvent logicielle qui agit sur un environnement. Par exemple, un « softbot » est un agent capable d'agir (par l'intermédiaire de commandes) sur des systèmes d'exploitation ou des systèmes de gestion de bases de données.

5 Wooldridge établit la liste des erreurs faites par les développeurs souhaitant réaliser des agents [Wooldridge, 1998]. La première cause qu'il met en avant est la surestimation des possibilités offertes par les agents

Si l'on s'en tient à cette définition « faible », les virus et démons UNIX sont des agents. La définition « forte » est d'avantage anthropomorphique. Les chercheurs considérant cette vision des agents leur attribuent des propriétés mentales comme la croyance, l'intentionnalité ou l'émotion.

La notion d'agent prenant toute sa signification dans la multiplicité, nous n'aborderons pas plus en détail ce concept et passerons donc aux systèmes multi-agents.

1.1.2. Systèmes multi-agents

Un système multi-agents est, selon [Bond et al., 1988] « un ensemble d'entités qui coordonnent leurs connaissances, buts, expériences et plans pour agir ou résoudre des problèmes, incluant le problème de la coordination inter-agents lui-même ». Briot [Briot et al., 2001] définit simplement un système multi-agents comme « un ensemble organisé d'agents », partant de la définition d'un système comme un « ensemble organisé d'entités ».

Des agents forment un système parce qu'ils évoluent dans le même environnement mais également car ils interagissent entre eux. Ces systèmes peuvent être qualifiés d'ouverts ou de fermés selon que les agents peuvent ou non y entrer et en sortir librement. Les SMA peuvent être homogènes c'est à dire composés d'agents identiques ou hétérogènes. L'approche « VOYELLES » [Demazeau, 1995] permet d'étudier les SMA à la fois au niveau de l'agent et du système complet. Pour cela elle met en avant les cinq facettes A,E,I,O,U des SMA : les Agents, leur Environnement, leurs Interactions, leurs Organisations et les Utilisateurs du système.

Outre ces cinq facettes que nous décrirons en détail ultérieurement, l'approche VOYELLES définit les propriétés importantes des SMA qui sont : la distribution, l'émergence, le principe de récursion, l'autonomie et la délégation.

Distribution

Un système multi-agents est composé d'entités. Dans un SMA, la distribution concerne, en particulier, les connaissances et les capacités d'action et de décision. L'une des problématiques inhérentes à la conception des SMA est de guider les choix permettant d'aboutir à cette distribution.

Émergence

L'émergence est un concept complexe relatif aux systèmes complexes. [Drogoul, 1993] en donne la définition suivante :

> *« L'émergence est un concept flou, mal défini et souvent décrié. Il dénote d'une manière générale, dans les systèmes qualifiés de complexes ou de non-linéaires, l'apparition d'un effet global qui n'est pas déductible de la connaissance des causes locales. Dans un système multi-agents, il dénote donc l'apparition d'un phénomène global qui n'est pas explicitement programmé dans le comportement des agents »*

Conformément à cette vision, on définit les fonctionnalités d'un SMA comme la somme des fonctionnalités des entités qui le composent et des fonctionnalités émergentes. Ce que résume [Demazeau, 2003] par cette formule.

F(SMA) = Σ(F(Entités)) + Fonctionnalités Emergentes.

F(X) dénotant les fonctionnalités du système X.

Récursion

Ce principe stipule qu'une entité composant un SMA peut, elle même, être composée de plusieurs entités. Par exemple, dans une société d'agents, une organisation peut, elle même, être incluse dans une organisation plus importante. De même, comme l'illustre la figure I.1, un système multi-agents peut être considéré, à un niveau d'abstraction plus élevé, comme un unique agent.

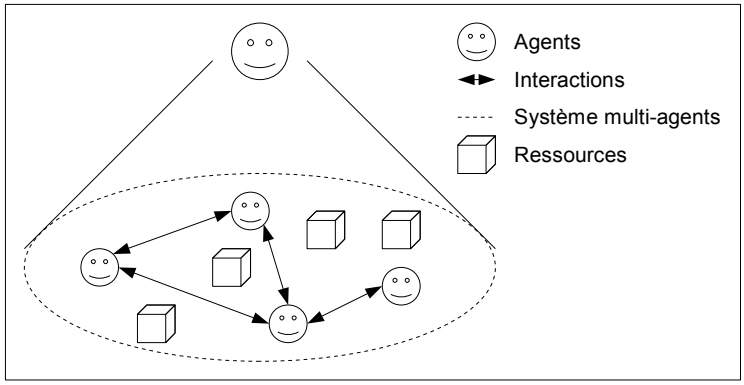

Figure I.1 : *Principe de récursion.*

Délégation

La délégation est une propriété importante des SMA. Un agent peut accomplir un ensemble de tâches à la demande d'un utilisateur ou d'un autre agent.

Autonomie

L'autonomie est l'une des principales propriétés des SMA qui les rend intéressants vis à vis de notre application de piétons virtuels. La notion d'autonomie est relative et a été discutée (entre autres) par [Steels, 1995] [Castelfranchi, 1995] et [Carabela et al., 2003]. Pour Carabela « Un agent X est autonome par rapport à Y dans un contexte C, si, dans C son comportement n'est pas imposé par Y ». Suivant les cinq facettes de VOYELLES il distingue différentes autonomies :

- U-autonomie : un agent est autonome vis à vis de l'utilisateur s'il peut réaliser certaines tâches sans son intervention et sans son autorisation.
- I-autonomie (ou autonomie sociale) : un agent est socialement autonome s'il peut refuser d'interagir avec les autres agents.
- O-autonomie (Norme- autonomie) : une société étant régie par des règles, un agent est O-autonome s'il peut décider de ne pas respecter ces règles.
- E-Autonomie : Un agent est E-Autonome, s'il n'est pas dirigé par son environnement. Cette hypothèse est généralement prise dans les SMA car un agent est simplement influencé par son environnement.
- A-Autonomie. Un agent est autonome par rapport à lui même s'il a plusieurs comportements disponibles et qu'il peut choisir lequel il souhaite adopter.

Dans tous les cas, il existe des degrés d'autonomie. Il n'est, par exemple, pas souhaitable qu'un agent soit totalement autonome vis à vis de l'utilisateur ; Carabela propose alors le concept d'autonomie adaptable. De même, il existe des situations particulières qui conduisent un agent à violer une règle (sciemment ou non) pour atteindre ses objectifs.

Les piétons virtuels sont autonomes dans les cinq sens du terme. Ils doivent être capables de prendre des décisions sans intervention extérieure d'un utilisateur. Ils peuvent refuser d'interagir avec les autres piétons, par exemple en refusant de leur céder le passage. Ils ne respectent pas nécessairement les règles du code de la route. Ils agissent comme ils le souhaitent dans leur environnement, tout en respectant certaines contraintes (ils ne peuvent pas passer au travers d'un obstacle).

Nous détaillerons dans les paragraphes suivants les cinq facettes A,E,I,O,U des SMA : les Agents, leur Environnement, leurs Interactions, leurs Organisations et les Utilisateurs du système.

1.2 Agents

Pour être autonome, l'agent devra être capable de réagir aux changements de l'environnement et de gérer ses interactions. La figure I.2 montre une vison synthétique de l'agent qui met en avant trois fonctions principales : la perception, le raisonnement et l'action.

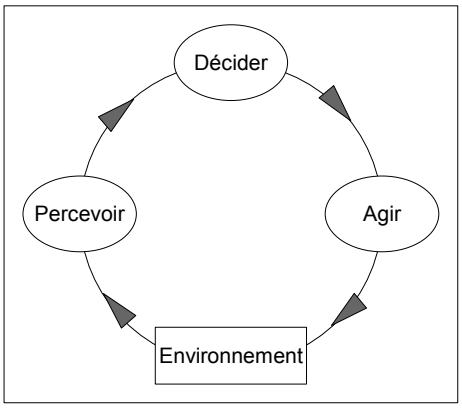

Figure I.2 : *Vision synthétique d'un agent.*

L'agent peut mettre en oeuvre une perception active ou passive. La perception est dite active (ou volontaire) lorsque la recherche d'informations est dirigée par les buts de l'agent. Elle s'oppose à la perception passive qui consiste à recueillir toutes les informations disponibles contenues dans l'espace de perception.

La manière dont un agent agit sur son environnement dépend des applications et nous n'entrerons pas dans les détails pour l'instant. Notons que la modélisation de l'action est un problème en soi [Ferber, 1995].

L'agent doit résoudre le délicat problème de la sélection d'actions. C'est sur ce point central que les agents sont classés en deux principales catégories : les agents réactifs et les agents cognitifs. L'approche réactive [Minsky, 1988] [Brooks, 1986], est fondée sur l'hypothèse qu'il n'est pas nécessaire que chaque agent soit intelligent pour parvenir à un comportement global intelligent. Les agents réactifs ne possèdent donc pas de mémoire et agissent de manière réflexe. Les agents cognitifs possèdent une mémoire (croyances) et peuvent utiliser des mécanismes de raisonnements symboliques complexes. Les agents BDI [Rao et al., 1991]

[Mandiau, 1993] sont l'un des exemples les plus significatifs. Le tableau I.1 tiré de [Reichgelt, 1990] résume les différences entre agents réactifs et cognitifs.

Tableau I.1 : *Comparaisons des agents réactifs et cognitifs [Reichgelt, 1990].*

Systèmes d'agents cognitifs	***Systèmes d'agents réactifs***
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agents complexes	Fonctionnement stimulus / réponse
Petit nombre d'agents	Grand nombre d'agents

Les propriétés réactives et cognitives étant complémentaires, il existe depuis longtemps de nombreux agents hybrides (entre autres [Fergusson, 1992]) qui utilisent des techniques issues de ces deux approches. Il n'y a pas donc de réel clivage entre les agents comme le laissait entendre le tableau I.1 ; la figure I.3 [Ferber, 1995] montre un continuum s'étendant entre deux extrêmes. Un tour d'horizon des architectures d'agent est présenté dans [Müller, 1997].

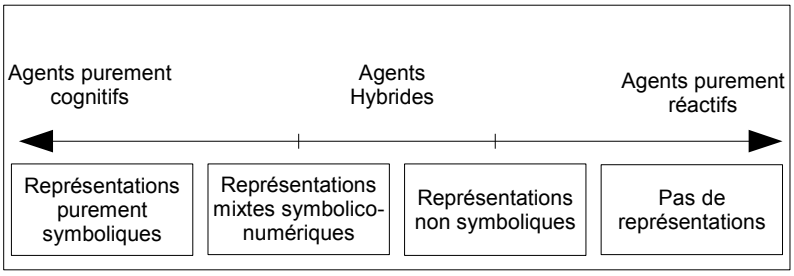

Figure I.3 : *Distinction cognitif / réactif [Ferber, 1995].*

Qu'il soit symbolique ou non, le raisonnement d'un agent doit lui permettre d'agir de manière rationnelle et de réagir aux changements de son environnement.

1.3 Environnement

Les agents sont plongés dans leur environnement constitué d'objets sur lesquels ils peuvent agir ou qui contraignent leurs actions. Par exemple, des piétons virtuels sont immergés dans leur environnement urbain simulé. Ils peuvent appuyer sur un bouton de feux, mais ne peuvent pas traverser un mur.

Dans le cadre des SMA, on distingue couramment l'environnement d'un agent et l'environnement d'un SMA. L'environnement d'un agent est constitué des objets (environnement physique) et des autres agents (environnement social) du système auquel il appartient. L'environnement d'un SMA est constitué de tout objet ou agent qui n'appartient pas au système.

Un environnement possède de nombreuses propriétés, pour [Russell et al., 1995] il peut être continu ou discret, déterministe ou non, dynamique ou statique, accessible ou inaccessible.

Un environnement est qualifié de discret (par opposition à continu) si l'ensemble des perceptions et l'ensemble des actions possibles sur cet environnement sont finis. Le monde réel est continu. Pour être le plus réaliste possible, l'environnement urbain simulé doit être continu.

L'environnement d'un système est qualifié de déterministe si une action du système sur cet environnement a un effet unique et certain. L'environnement d'un acteur virtuel peut donc être non déterministe : si deux acteurs virtuels agissent au même instant sur un même objet, les acteurs virtuels ne peuvent prédire la réaction de l'objet.

L'environnement d'un système est qualifié de statique (par opposition à dynamique) si son état ne dépend que de ses états antérieurs et des actions réalisées par le système. L'environnement physique d'un agent est donc le plus souvent dynamique car d'autres agents y agissent. L'état d'un environnement dynamique ne peut pas être prédit par le système.

L'environnement d'un système est accessible si ce système peut, à chaque instant, le percevoir de manière complète et précise. A titre d'exemple, l'environnement d'un robot n'est pas complètement accessible car ses capteurs possèdent une précision et une portée limitée. Dans un environnement simulé ce problème d'accessibilité est plus complexe, nous reviendrons sur ce point dans la seconde section du chapitre.

L'existence d'interactions entre les agents constitue la principale différence avec l'IA « classique », c'est pourquoi l'interaction est une des briques de VOYELLES.

1.4 Interactions

Pour [Ferber, 1995], « une interaction est une mise en relation de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques ». Plus simplement, il y a interaction entre des agents si leurs comportements sont influencés par les actions des autres.

[Ferber, 1995] distingue huit types de situations d'interaction dont les trois principales catégories sont l'indifférence, la coopération ou l'antagonisme. Dans le cadre des piétons virtuels les trois formes d'interactions sont envisageables. Par exemple, il y a indifférence si les personnages se croisent sur un trottoir et qu'il y a assez de place ; coopération si les personnages essaient de coordonner leurs mouvements pour se croiser dans un passage étroit ; ou antagonisme s'ils ne font preuve d'aucune courtoisie.

Un autre élément lié à l'interaction est la communication. Les agents peuvent interagir : sans communiquer [Rosenschein, 1985], en utilisant un support de communication (un tableau noir (Hearsay-II [Erman, 1980]), ou l'environnement [Drogoul, 1993]), en envoyant des messages simples ou en utilisant un protocole de communication sophistiqué.

Il existe de nombreuses formes d'interactions. Chacune d'entre elles constituant un sujet de recherche à part entière, il n'est pas possible d'entrer dans les détails. Comme toute société, un SMA est fondé sur un ensemble de règles régissant les interactions entre ses membres. L'organisation d'un SMA permet de définir un cadre pour ces interactions.

1.5 Organisation

L'organisation peut être vue comme la conséquence des interactions entre agents ou réciproquement comme une manière de contraindre ou de simplifier les interactions. Selon [Morin, 1977] cité dans [Ferber, 1995], « Une organisation peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus. »

Une structure organisationnelle peut être prédéfinie puis appliquée à une organisation concrète. Une structure organisationnelle peut posséder différents niveaux et même être définie récursivement [Adam, 2000]. Les organisations peuvent être utilisées dans les SMA

afin de structurer les interactions : structures d'autorités qui définissent « qui dirigent » ou structurent de responsabilités qui définissent « qui peut faire quoi ». Une organisation permet, par exemple, d'assigner des rôles aux agents : consommateur / producteur, coordinateur / exécutant, prédateur / proie... (cf. Agent / Groupe / Rôle [Gutknecht et al,1998]), de définir des redondances, etc. Statiques dans les premiers SMA, les organisations sont devenues dynamiques afin de s'adapter au contexte, passant par exemple d'hétérarchique à hiérarchique lorsque la tâche à accomplir requiert une certaine coordination, cf. [Le Strugeon, 1995].

A l'inverse, une organisation peut également être émergente [Drogoul, 1993] c'est à dire observable *a posteriori,* comme dans les systèmes d'agents réactifs. Dans ce cas, les SMA peuvent être utilisés pour simuler et comprendre des organisations complexes (colonies d'insectes sociaux dans le cadre éthologie).

1.6 Utilisateurs

Depuis quelques années, l'approche VOYELLES intègre la prise en compte des utilisateurs. Oubliés dans un premier temps, vus comme de simples « clients » par la suite, l'avenir des SMA est, selon [Demazeau, 2005], dans la prise en compte des utilisateurs comme de réels partenaires.

La bibliographie sur ce domaine est importante et les applications d'agents assistants, de recherches d'information, de personnalisation de l'information [Rozé, 2003] sont nombreuses. Se référer à, par exemple, [Grislin et al., 2001] pour plus de détails.

Dans un contexte proche de l'animation, il convient de citer les travaux de [Sansonnet et al., 2006] et du Groupe ACA (Agents Conversationnels Animés) du collège SMA.

Avant de conclure ce tour d'horizon des systèmes multi-agents, nous proposons un bref aperçu des applications pour lesquelles ils sont utilisés.

1.7 Domaines d'application des systèmes multi-agents

Les applications des SMA sont nombreuses [Mandiau et al., 2002]. La liste des applications que nous présentons est inspirée de la classification de [Ferber, 1995].

Résolution de problèmes

Historiquement ce sont les premières applications des SMA. Il est possible de résoudre un problème de manière distribuée, notamment en demandant à plusieurs experts de se pencher sur différents aspects d'un même problème.

Génie logiciel

Les agents peuvent être vus comme une méthode de conception et de programmation de logiciels (au même titre que l'orienté objet). Dans ce cadre, les SMA permettent de développer des logiciels particulièrement adaptables.

Robotique distribuée

Les principales applications sont les systèmes multi-robots. Les robots fourrageurs en sont un exemple, au même titre que la « Robot cup », coupe de football entre des équipes de robots. Le programme ATRONS [Jørgensen et al., 2004] développe de petits robots simples, capables de s'accrocher entre eux et de former des structures modulables.

La simulation multi-agents

La simulation multi-agents a apporté une solution nouvelle au concept de modèle et de simulation. En simulant les entités qui composent le système et les interactions entre elles, les modèles deviennent microscopiques et non plus macroscopiques. Cette possibilité permet d'essayer de modéliser des systèmes complexes, en modélisant les relations entre les entités et en observant les répercussions sur la globalité du système.

Dans cette partie nous avons essayé de présenter les systèmes multi-agents en suivant les différentes facettes mises en avant par la méthode VOYELLES [Demazeau, 2005]. Notons que dans la pratique un SMA développe rarement toutes les facettes. Nous avons finalement présenté les applications des SMA. La simulation multi-agents est un domaine d'application dans lequel on peut inclure l'animation comportementale. Les environnements virtuels (dont les jeux vidéo) ont toujours été un domaine d'application privilégié pour l'intelligence artificielle. Selon John Laird cité dans [Robert, 2001] et [Mathieu et al., 2003], les jeux vidéo sont un excellent cadre d'application pour l'IA. Ils proposent, en effet, une alternative intéressante entre les travaux de recherche spécialisés et le monde (physique) réel très complexe, nécessitant des compétences multiples et des investissements importants. Cette rapide introduction aux agents nous permet de mieux introduire les environnements et les acteurs virtuels dans la partie suivante. En effet, les SMA apportent un cadre ainsi que des solutions particulièrement adaptées à ce domaine.

2. Apport des SMA pour les acteurs virtuels

Après avoir introduit le concept d'animation comportementale et présenté l'architecture d'un acteur virtuel, nous verrons comment les principes des SMA sont appliqués dans ce domaine. Pour cela, nous aborderons les différentes facettes de l'approche « VOYELLES » [Demazeau, 1995], d'ailleurs reprise par [Tisseau, 2001] dans le cadre de la réalité virtuelle. Nous aborderons donc la modélisation de l'Environnement, des Utilisateurs, des Interactions, des Organisations et des Agents.

2.1 L'animation comportementale

2.1.1. Définition

Notre travail ne concerne pas directement l'animation, néanmoins une brève incartade dans ce domaine est nécessaire afin de définir le concept d'animation comportementale. Selon B. Arnaldi et G. Hégron cités dans [Moreau, 1998], l'animation comportementale concerne à la fois : les modèles de transformations internes (croissance de plantes) et modèles de transformations externes qui agissent sur leur environnement (animation d'objets ou de personnages). Nous n'aborderons pas les modèles de transformations internes.

Réaliser des animations suppose bien sûr de modéliser des mouvements. Les modèles utilisés en animation s'organisent en trois classes. On parle couramment des modèles générateurs, descriptifs et comportementaux.

- Les modèles descriptifs décrivent les mouvements par leurs effets. Une équation paramétrique peut, par exemple, être utilisée afin de décrire le mouvement d'un objet.
- Les modèles générateurs sont plus sophistiqués puisqu'ils décrivent les causes du mouvement. Ils se basent donc le plus souvent sur des modèles mécaniques (principe fondamental de la dynamique).
- Les modèles d'animations comportementaux s'appuient sur les modèles descriptifs et générateurs. Ce ne sont pas des modèles de mouvements à proprement dits : ils apportent un niveau d'abstraction plus élevé. Ils ont pour but de donner de l'autonomie à l'objet [Fluchs et al., 2003].

L'autonomie n'est pas l'objectif ultime de l'animation comportementale[6], mais c'est bien ce qui fait la différence avec les autres modèles d'animation. Les poissons de [Terzopoulos et al, 1994] sont l'un des exemples les plus significatifs du domaine. Leur modèle d'animation fait appel à un modèle générateur car les forces qui sont à l'origine du mouvement ont été modélisées. La couche d'animation comportementale apporte les algorithmes d'apprentissage nécessaires à la coordination des mouvements. En quelque sorte ces poissons apprennent eux-même à nager. Pour les acteurs virtuels, l'autonomie concerne également le comportement.

2.1.2. Les acteurs virtuels

La spécificité de l'animation comportementale réside dans l'autonomie apportée aux modèles d'animation. Pour les acteurs virtuels, cette autonomie concerne non seulement leurs

6 Voir [Blumberg et al., 1995] : Les acteurs virtuels ont souvent vocation à interagir avec l'utilisateur. Leur intérêt est limité s'ils ne se préoccupent pas de lui.

mouvements mais également leurs comportements. Pina a établi une classification des acteurs virtuels [Pina et al., 2000] qui est synthétisée dans le tableau I.2. Celle-ci illustre leur évolution, depuis le clone dirigé par un utilisateur et reproduisant les mêmes animations jusqu'à l'acteur autonome capable d'adapter son comportement et ses mouvements.

Tableau I.2 : *Classification des acteurs synthétiques [Pina et al., 2000].*

Classe d'acteur synthétique	***Comportement répliqué***	***Comportement modélisé***	***Mouvements répliqués***	***Mouvements modélisés***
Pur avatar ou clone	X		X	
Acteur indépendant non limité	X			X
Acteur autonome		X	X	
Acteur autonome non limité		X		X

La figure I.4 [Blumberg et al., 1995] présente l'architecture globale d'un acteur virtuel permettant de lier comportement et mouvement. Cette architecture a été utilisée dans de multiples projets (par exemple Improv [Perlin et al., 1996]). Elle décompose les acteurs virtuels en trois modules principaux qui prennent respectivement en charge les aspects : graphique, animation et comportement. Le module « comportement » commande l'exécution de mouvements complexes au « système moteur ». Celui-ci contrôle les angles des articulations du personnage à chaque instant de la simulation. Les changements sont ensuite appliqués à la géométrie du personnage pour être finalement affichés. Nous présentons maintenant chacun des modules mis en avant sur la figure I.4.

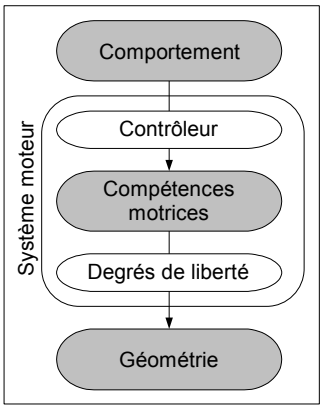

Figure I.4 : *Architecture de personnage virtuel [Blumberg et al.,1995]*

Module graphique

Même s'il n'est pas anodin, le module graphique d'un acteur virtuel ne sera pas abordé en détail. La figure I.5 ci-dessous montre un personnage du projet RESPECT. Le modèle graphique est un modèle surfacique sur lequel des textures ont été appliquées.

Figure I.5 : *Personnage dans RESPECT.*

Degrés de liberté

Un acteur virtuel n'est pas seulement défini par son apparence extérieure. Il possède une sorte de squelette formé par un solide articulé. La figure I.6 [Grislin et al., 2002] en est un exemple. Les segments corporels (représentés par des rectangles) sont reliés entre eux par des liaisons mécaniques (représentées par des lignes) qui modélisent les articulations du personnage. Celles-ci sont de trois types : pivot, cardan ou rotule et correspondent respectivement à 1, 2, ou 3 degrés de liberté (DDL) soit autant de rotations.

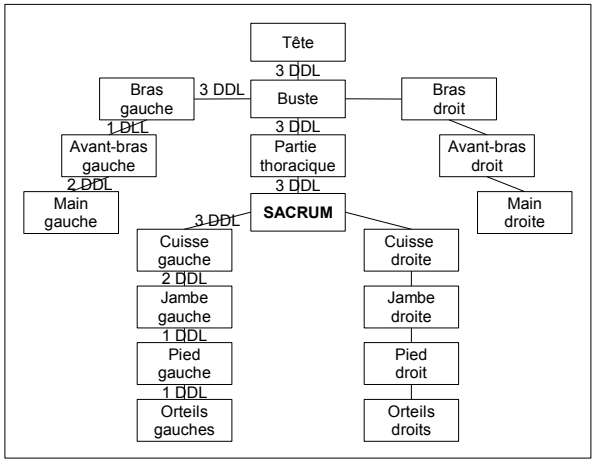

Figure I.6 : *Squelette hiérarchique d'un personnage dans RESPECT [Grislin et al., 2002].*

Pour animer le personnage, le module « compétences motrices » fait varier les angles d'articulation du squelette. Les degrés de liberté constituent donc une interface entre graphisme et compétence motrice.

Compétences motrices

Le module « compétences motrices » constitue le coeur du système moteur d'un acteur virtuel (cf. figure I.4). Parmi les compétences motrices retenues dans le cadre de RESPECT nous pouvons citer les suivantes : marcher, courir, tourner la tête ou appuyer sur le bouton d'un feu. Tourner la tête du personnage correspond à un mouvement simple puisqu'il s'agit de faire varier un seul angle du squelette hiérarchique. Les mouvements comme la marche ou la course mettent en jeu beaucoup de degrés de liberté et sont donc particulièrement complexes. Dans le cadre de RESPECT, ces mouvements ont donc été enregistrés.

Le contrôleur : Interface comportement / Système moteur

Interface entre le module « comportement » et les « compétences motrices », le contrôleur est un module particulièrement délicat à concevoir. Reynolds [Reynolds, 1999] prétend que cette interface permet de réaliser le module comportement sans se préoccuper du corps du personnage. Le principal problème que ce genre de module essaie de résoudre concerne les transitions entre les différents mouvements (ce que les animateurs appellent le « blending »).

Comportement

Simuler le comportement d'un personnage virtuel est un problème très ardu [Thalmann, 2003]. Le module responsable du comportement commande l'exécution de mouvements au système moteur. Pour que le comportement du personnage soit crédible, il faut produire une suite logique de mouvements afin d'atteindre les objectifs fixés par l'animateur tout en prenant en compte les évolutions de la simulation. Pour y parvenir, les chercheurs ont, dans un premier temps, développé des langages de scripts puis ils se sont tournés vers les méthodes utilisées en intelligence artificielle. Ce point sera développé en partie 3.6.

Ayant présenté l'architecture d'un acteur virtuel, nous traiterons à présent des environnements virtuels dans lesquels ils évoluent.

2.2 Modélisation des environnements virtuels peuplés

Les images de synthèse sont à l'origine des environnements virtuels. Afin « d'accueillir » les acteurs virtuels, les environnements ont été structurés et « informés » afin de devenir une source d'informations pour les agents et de supporter leurs actions. Les techniques de modélisation « agent » ont permis de modéliser la dynamique de ces environnements.

[Tecchia et al., 2000] propose de discrétiser l'espace en utilisant des grilles pour modéliser l'environnement. Chaque acteur occupe une case tandis que les obstacles en occupent plusieurs. La discrétisation permet de simplifier les algorithmes qui permettent aux acteurs de percevoir l'environnement et de calculer leurs déplacements. Néanmoins, l'intérêt principal de ce travail est l'ajout d'informations sur cette grille. Elles facilitent l'adaptation du comportement en fonction de la case où l'acteur se trouve.

La ville est un environnement fortement structuré. Les travaux de [Farenc et al., 1998] ou [Thomas, 1999] ont eu pour but de modéliser ce type d'environnement. Leurs modèles sont basés sur cette structure ainsi que sur une approche orientée objet. Ils présentent deux avantages. Le premier apport est l'ajout d'informations concernant la géométrie de l'environnement. Par exemple, les voies de circulation sont modélisées par une axiale qui définit un repère curviligne. Les objets et acteurs virtuels sont positionnés par rapport à ce repère cuviligne. Ces informations permettent de simplifier les algorithmes de déplacement des personnages.

Le second avantage de ces deux modèles concerne l'introduction d'informations sémantiques et topologiques. La modélisation des voies inclut, par exemple, des informations comme le type d'usager, les limites de vitesses, etc. Les graphes topologiques de l'environnement décrivent les liens entre les voies connexes, très utiles pour planifier un déplacement à l'échelle de la ville. [Farenc et al., 1998] propose le concept d'entité environnementale (ENV) qui permet de structurer hiérarchiquement l'environnement. Une ENV représente un volume ou une surface. Une ENV peut, elle-même, être composée de

différentes ENV. Chaque ENV contient des informations comme la liste des objets qu'elle comporte et des informations sur les comportements associés à ce lieu. La structuration de l'environnement facilite sa création et permet de limiter la complexité des algorithmes de perception (cf. paragraphe 3.6).

Kalmann [Kalmann et al., 2000] introduit le concept d'objet intelligent (« smart object »). Un objet intelligent « encapsule » toutes les données et méthodes nécessaires à sa manipulation. Il inclut une machine d'état dont les transitions décrivent les interactions qu'il supporte et dont les états expriment ses configurations possibles (ouvert, fermé...). Il comporte également le code des animations correspondant à sa manipulation. Grâce à cette méthode, le système est réellement ouvert puisqu'il est possible d'ajouter des objets nouveaux sans que cela suppose de modifier les acteurs. La figure I.7, ci-dessous, illustre le concept. Les tiroirs sont des « smart objects » et décrivent le mouvement que doit effectuer la main du personnage pour ouvrir un tiroir. Grâce aux calculs de cinématique inverse, le personnage parvient à ouvrir n'importe lequel.

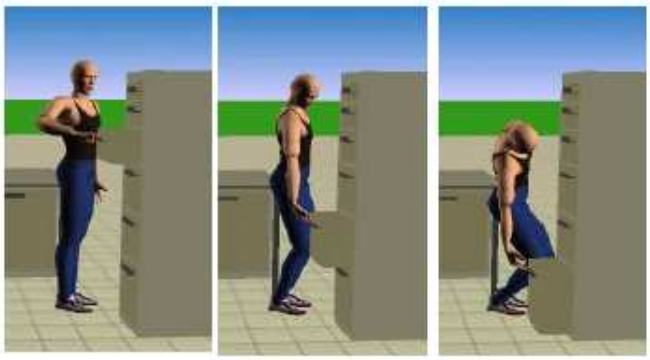

Figure I.7 : *Interaction avec un « smart-object »[Kalmann et al., 2000].*

Dans MASCARET (*MultiAgent System for Collaborative and Adaptative Realistic Environnement for Training*), la modélisation de l'environnement [Querrec, 2002] est particulièrement originale. Dans l'application présentée, l'environnement physique est en évolution constante. Ceci a conduit l'auteur à utiliser, non plus des objets, mais des agents réactifs pour le modéliser. Ces agents maintiennent une liste des voisins afin de simplifier le calcul de l'évolution de l'environnement (propagation de feu, déplacement de nuages de gaz, etc).

Le tableau I.3 résume les méthodes développées pour modéliser les environnements ainsi que leurs impacts sur les acteurs virtuels.

Tableau I.3 : *Impact de la modélisation de l'environnement sur les acteurs synthétiques.*

Auteurs	***Modélisation de l'environnement des acteurs virtuels.***	***Impacts positifs sur les acteurs virtuels***
[Farenc et al., 1998] [Thomas, 1999]	Structuration hiérarchique	Limitation des recherches en perception Accroissement de l'efficacité de la perception
	Structuration topologique	Simplification de la planification des déplacements
	Modélisation géométrique pertinente	Simplification des fonctions de navigation
	Informations sémantiques associées aux lieux simulés	Adaptation du comportement des agents à la nature des lieux.
[Kalmann et al., 2000]	Objets intelligents	Accroissement des possibilités d'interactions entre acteurs et objets Ouverture du système
[Querrec, 2002]	Agents réactifs	Calcul de l'évolution de l'environnement Interactions acteur / environnement

Pour conclure cette présentation des avancées en matière de modélisation de l'environnement, il faut rappeler que l'ajout de connaissances exploitables par les agents constitue un pas vers la richesse de leurs comportements mais n'impose aucune restriction sur les agents ou sur leurs comportements.

2.3 Utilisateurs d'environnements virtuels

Un avatar constitue l'interbrique E-U (environnement / utilisateur). C'est une représentation de l'utilisateur dans la simulation (cf. figure I.8). Les progrès accomplis en réalité virtuelle ont permis de réaliser une multitude d'interfaces, sensorielles (lunettes immersives), motrices (gants de données) et sensori-motrices (manette à retour de force). Ce matériel permet à l'utilisateur de se sentir immergé dans la simulation et d'y agir de manière « naturelle ».

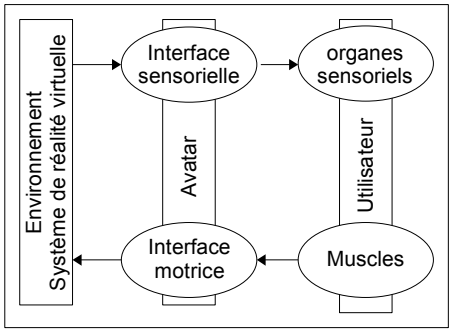

Figure I.8 : *Utilisateur d'un système de réalité virtuelle.*

Les exemples d'interactions évoluées entre acteurs virtuels et utilisateurs sont assez peu nombreux à notre connaissance. L'équipe « *Character* » du MediaLab (MIT) a réalisé quelques démonstrations dans lesquelles l'utilisateur peut interagir avec des personnages virtuels dont, par exemple, le chien *Silas* [Blumberg et al., 1995]. Des interfaces originales ont également été réalisées comme les « *stuffed animals* » (peluches bardées de capteurs) du projet « *Swamped* » (pour plus de détails voir http://characters.media.mit.edu).

Nous avons rapidement présenté l'avatar, moyen d'interaction entre l'utilisateur et les éléments du monde virtuel. Nous allons, à présent, voir les formes d'interaction entre les entités.

2.4 Interactions entre acteurs virtuels

Les acteurs sont souvent autonomes vis à vis de l'utilisateur. Il arrive néanmoins que ce dernier ait besoin de les contrôler. [Blumberg et al., 1995] ou [Caicédo et al., 2000] proposent plusieurs niveaux de contrôle qui consistent soit à ordonner de faire un mouvement soit à modifier une croyance.

Les situations d'interactions entre acteurs virtuels peuvent être nombreuses : compétition [Girard et al., 2001], collaboration [Querrec, 2002], indifférence, *etc*. Des interactions « physiques » entre les acteurs (comme se serrer la main) ont été développées [Rezzonico et al., 1995]. Les acteurs virtuels peuvent également communiquer par messages [Caicédo et al., 2000] ou de manière non verbale. Dans le cadre de la simulation de trafic routier (proche de l'animation comportementale) les agents doivent coordonner leurs actions [Champion, 2003] [Doniec et al., 2006]. [Caicédo et al., 2000] développe la notion de confiance entre les personnages qui permet de moduler leurs interactions. Les organisations permettent également de moduler les interactions.

2.5 Organisations des sociétés d'acteurs virtuels

Bien qu'on assigne souvent des rôles aux acteurs virtuels lors des démonstrations, les exemples d'organisations d'acteurs virtuels sont assez rares. [Caicédo et al., 2000] propose de grouper les personnages en familles disposant de capacités similaires. Dans MASCARET [Querrec et, al 2001] [Querrec, 2002] nous avons déjà cité l'exemple des agents réactifs qui modélisent l'environnement et maintiennent des relations d'accointance avec leurs voisins. Il faut également citer les agents cognitifs qui forment des organisations sociales. Ils sont groupés en équipes et réalisent les objectifs fixés par leurs supérieurs dans l'organisation.

[Reynolds, 1987] modélise des bancs de poissons ou des nuées d'oiseaux à partir d'agents réactifs simples. En effet, ses nuées constituent des groupes cohérents d'agents bien qu'elles ne soient composées que d'agent réactifs simples, ne communiquant pas et n'ayant pas conscience d'appartenir à un groupe.

Les organisations peuvent servir à simplifier les interactions entre agents. Lorsqu'elles sont inexistantes comme dans le cadre des acteurs virtuels, il n'y a que leurs modèles décisionnels qui peuvent les sortir de situations de conflits.

2.6 Comportement des acteurs virtuels

Dans le paragraphe 3.1.2 nous avons décrit l'ensemble de l'architecture d'un acteur virtuel, nous abordons à présent les détails concernant le module responsable de son comportement. Pour illustrer notre propos, nous nous appuierons sur le modèle C4 de l'équipe *character* du Medialab au MIT. C'est en effet un modèle éprouvé : de Silas [Blumberg et al., 1995] à Duncan [Isla et al., 2001] les membres de cette équipe ont développé de nombreux acteurs virtuels afin de tester et présenter leurs idées. C4 est bâti sur le modèle modulaire présenté sur la figure I.9. Ce modèle permet de tester différentes implémentations pour chaque module et d'en ajouter de nouveaux. La figure présente la version minimale du modèle. Les modules « mémoire » et « tableau noir » permettent de lier les modules principaux qui sont : la perception, la sélection d'actions, le système de navigation et le système moteur.

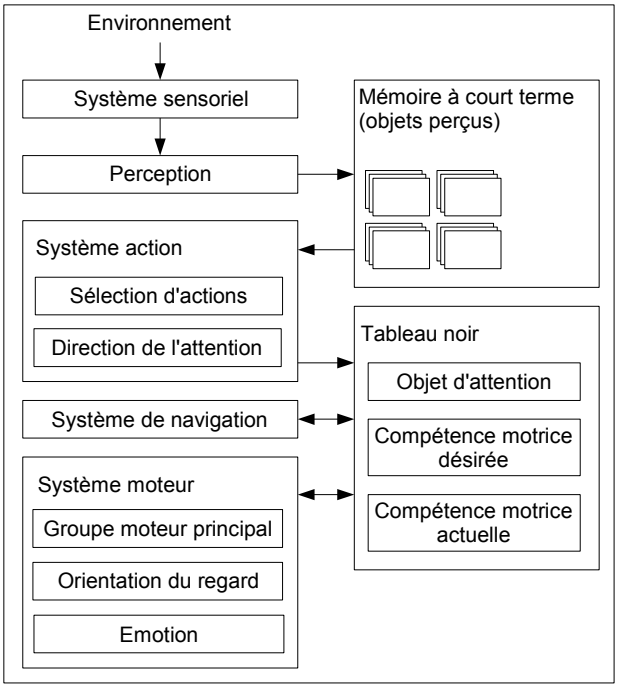

Figure I.9 : *Le modèle C4 [Isla et al., 2001].*

Présentons rapidement les modules principaux du modèle C4 (figure I.9) avant d'entrer dans les détails.

- La perception constitue l'interface entre l'environnement et l'agent. Ici les signaux captés par le système sensoriel sont identifiés par la perception qui leur assigne une signification.
- La mémoire est alimentée par la perception. Elle peut avoir son importance pour certaines applications.
- La sélection d'actions utilise la mémoire pour déterminer ce que l'agent va faire et sur quoi il va focaliser son attention.
- Le système de navigation est responsable de la gestion des déplacements. Les auteurs ont choisi de le découpler de la sélection d'actions mais il peut y être associé.
- Le système moteur (cf. paragraphe 3.1.2) est responsable du bon enchaînement des mouvements. Ici, les auteurs ont choisi de différencier les mouvements principaux (marche...) et l'orientation du regard (tête et les yeux), l'émotion a un impact sur la réalisation des mouvements (démarche plus ou moins assurée par exemple).

Nous détaillerons en particulier la perception, la sélection d'actions et la navigation.

2.6.1. Perception des acteurs virtuels

La perception est l'interface entre l'environnement et l'agent (interbrique A-E). Nous proposons dans ce point de rappeler pourquoi la perception est importante et comment les systèmes de perception sont développés.

En réalité virtuelle, les informations perçues sont des données associées aux objets de l'environnement. Dans ce contexte, toutes les informations nécessaires sont disponibles et stockées dans la structure de données qui décrit l'environnement et permet de générer les images (cf. paragraphe 3.2). Pourquoi aborder la perception puisque les informations sont

disponibles et que les problèmes de mesures et d'interprétations ne se posent pas dans le cadre de l'animation comportementale ? Ne suffirait-il pas alors de parcourir la structure de données pour retrouver les informations nécessaires ? [Aylett et al., 2000] et [Noser et al., 1995] montrent que cette solution n'est envisageable que pour des mondes virtuels restreints car :

- parcourir la totalité de la structure de données décrivant l'ensemble du monde peut s'avérer long et donc pénalisant ;
- proposer trop d'informations au modèle décisionnel aboutirait à une explosion combinatoire des choix qui s'offrent au personnage virtuel et donc à des temps de raisonnement trop longs ;
- même si ces deux premiers points n'induisent pas de problèmes critiques, les êtres virtuels devenus omniscients n'auraient pas des comportements crédibles.

Pour limiter ces problèmes, une première solution est de structurer l'environnement (cf paragraphe 3.2). La structure offre la possibilité de réduire les tests de perception à une partie de l'environnement (comme un quartier ou une rue). L'ajout d'informations sémantiques et topologiques a également des répercussions positives sur la perception.

Le développement de « capteurs virtuels » apporte une seconde solution au problème de perception. Les capteurs virtuels sont les outils grâce auxquels le personnage virtuel reçoit les informations concernant le monde qui l'entoure. Ils opèrent en tant que « filtres » sur les informations disponibles afin de ne recueillir que celles portant sur les objets présents dans son champ de vision (dans le cas particulier de la vue). La modélisation de cette perception limitée permet de garantir l'honnêteté[7] de la perception. Les chercheurs font en général la supposition qu'une perception honnête rend l'acteur plus crédible. L'utilisation de ces capteurs limite les problèmes présentés précédemment et améliore la portabilité des agents. Ils apportent, en effet, une interface qui permet de réaliser une abstraction entre l'environnement et les mécanismes de raisonnement de l'agent.

La réalisation des capteurs virtuels peut prendre différentes formes. Outre le parcours de la structure de données que nous avons déjà évoqué, il existe des méthodes bien plus sophistiquées. Dans [Noser et al., 1995], la technique présentée consiste à effectuer le traitement de l'image (et du tampon de profondeur) de la scène calculée suivant le point de vue du personnage. Cette technique offre l'avantage de tenir compte des occultations entre les objets et rend la perception relativement indépendante de l'environnement. Mais elle est aussi complexe donc coûteuse en temps de calcul.

Il existe donc de nombreuses méthodes qui permettent de rendre la perception efficace et honnête. Certains projets comme [Blumberg et al., 1995] utilisent les différentes techniques afin de bénéficier de leurs avantages respectifs. Nous n'avons cité que des exemples concernant la vision, néanmoins des capteurs virtuels pour l'ouïe ou le toucher ont été développés [Noser et al., 1995].

La perception volontaire des acteurs virtuels est, à titre d'exemple, abordée dans le système Jack [Chopra-Khullar et al., 1999]. Pour chaque tâche réalisée par le personnage, le concepteur fournit une liste des objets à observer et la perception pointe les objets susceptibles de distraire le personnage (objets en mouvement, brillants, ou bruyants). L'agent dirige donc son regard vers ces différents objets.

7 L'« honnêteté » de la perception permet de garantir que le comportement de l'agent n'est pas influencé par des éléments qu'il n'est pas sensé percevoir.

Quelle que soit la technique utilisée, la perception fournit des informations à l'agent pour qu'il puisse prendre des décisions.

2.6.2. Modèle décisionnel des acteurs virtuels

Les applications de l'animation comportementale sont nombreuses et chaque modèle décisionnel répond souvent à un problème donné. De manière générale le modèle décisionnel d'un personnage doit lui permettre de :

- Agir de manière rationnelle (c'est à dire de façon à atteindre un objectif) tout en réagissant aux actions des joueurs et des autres personnages (de manière réflexe lorsque cela est nécessaire, en planifiant des actions lorsque cela est possible).
- Gérer ses déplacements en prenant en compte les autres personnages.
- (éventuellement) Gérer sa dimension sociale (autonomie, conflits, collaboration) envers les autres personnages ou les utilisateurs.
- (éventuellement) Posséder une personnalité. En effet, la personnalité d'un personnage, se reflète dans ses choix. Dans le cadre de RESPECT on peut imaginer des acteurs prudents ou pressés ... Leur personnalité peut, par exemple, s'exprimer dans le choix d'un lieu de traversée.

Un modèle décisionnel doit donc permettre à l'agent de sélectionner parmi un ensemble d'actions réalisables à un instant t, celle qui est la plus adaptée en fonction de l'état de l'environnement, de l'état de l'agent et de ses objectifs. La difficulté de ce problème réside dans le fait que les actions réalisables sont généralement concurrentielles et conflictuelles.

Comme nous l'avons déjà souligné, la conception des modèles décisionnels est fortement liée à l'application. Il est donc d'autant plus difficile de les évaluer ou de les comparer. Moreau a listé les principales caractéristiques d'un modèle décisionnel d'acteur virtuel [Moreau, 1998] que nous avons reprises dans le tableau I.4. Les principales caractéristiques mises en avant dans ce tableau sont la réactivité, la modularité et la concurrence, les autres contraintes dérivant de celles-ci. La modularité est importante car elle permet une mise à jour du modèle. (cf. [Sukthankar, 1997] qui a comparé des architectures monolithiques et modulaires.) La réactivité est nécessaire puisque les temps de réponse sont importants. Moreau entend par réactivité que le système est suffisamment rapide pour traiter les informations qu'il reçoit. En effet, le temps nécessaire au modèle décisionnel pour effectuer ses choix doit être maîtrisé afin d'être utilisable dans des applications temps réel. La concurrence doit permettre d'évaluer simultanément différentes possibilités.

Tableau I.4 : *Contraintes d'un modèle décisionnel d'acteur virtuel [Moreau, 1998].*

Contraintes	***Motivation***
Modularité	Mise à jour
Réactivité	Simulation interactive
Hiérarchie	Architecture de contrôle
Concurrence	Activités menées simultanément
Préemption	Résolution de conflits
Exceptions	Traitement des situations exceptionnelles
Gestion du temps	Activités cognitives de fréquences différentes
Flots de données	Communication entre les différents niveaux

[Duthen, 2002] propose de classer les modèles décisionnels en trois principales approches : impérative, déclarative et basée sur l'émergence. L'approche impérative utilise des scripts qui définissent le comportement *a priori*. L'approche déclarative correspond à l'utilisation d'agents cognitifs et l'approche basée sur l'émergence fait référence aux agents réactifs.

Approche impérative

La méthode impérative vise à décrire entièrement le comportement d'un agent. Il existe beaucoup d'outils de description explicite du comportement. Ceux-ci se basent souvent sur l'utilisation de langages de scripts, de langages réactifs ou d'automates. Ils ont donné naissance à des langages dédiés, lesquels sont ou seront utilisés par des programmes et interfaces graphiques utilisables par un public large [Richard, 2001].

Le système Improv [Perlin et al., 1996] propose un langage adapté à la description des comportements et des animations. L'introduction de probabilité entre les différentes possibilités décrites par le concepteur rend les personnages moins prévisibles.

L'utilisation d'automates à états finis est une manière simple de décrire un comportement. HPTS (Hierarchical Parallel Transition Systems) [Moreau, 1998] est un langage permettant la description de modèles décisionnels basés sur les automates. La hiérarchisation des automates permet d'adopter une démarche descendante et de décrire le comportement selon différents niveaux d'abstraction. La figure I.10 décrit une partie du comportement d'un conducteur. Des « fonctions d'intégration » permettent de prendre en compte l'activation simultanée de certains états proposant des actions contradictoires. La formulation de ces fonctions n'est pas toujours simple (cf. chapitre II).

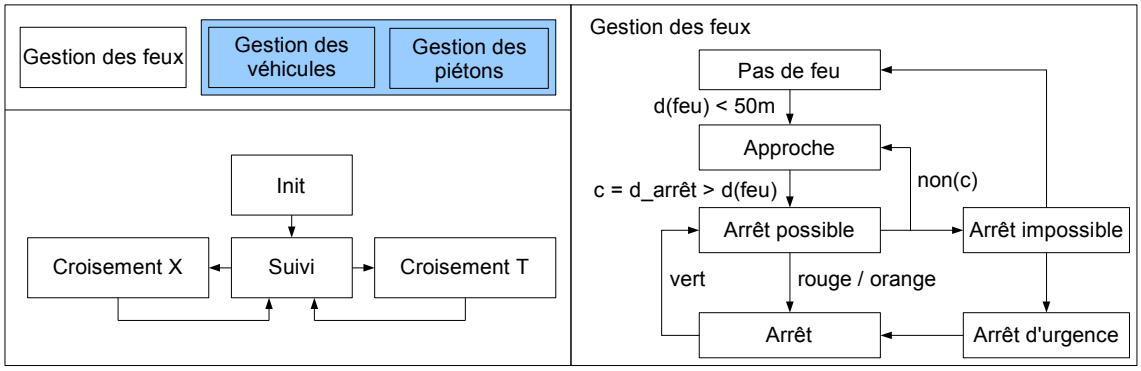

Figure I.10 : *selon [Moreau, 1998] : Comportement d'un conducteur, à gauche une vue de l'ensemble, à droite, le détail de l'automate de « gestion des feux ».*

Avec l'approche impérative, le concepteur donne une description complète du comportement du personnage. Le concepteur maîtrise donc totalement le comportement de son personnage, mais il doit considérer toutes les possibilités. Le défaut est que l'agent risque de mal réagir aux cas imprévus. C'est pour cette raison que cette approche ne nous semble pas adaptée aux environnements dynamiques.

Approche cognitive

L'utilisation de méthodes cognitives offre aux agents la possibilité de réaliser des raisonnements abstraits leur permettant de résoudre leurs problèmes. L'architecture utilisée par Caicédo [Caicédo et al., 2000] en est un exemple. Cet agent BDI (Beliefs, Desires, Intentions) est bâti autour d'un moteur d'inférences (cf. figure I.11) qui manipule des connaissances symboliques.

A chaque cycle, les croyances à court terme de l'agent sont remises à jour en fonction de ce qui est perçu et ce qui est oublié (les informations perçues sont mémorisées et datées, elles sont supprimées après un certain temps). Les croyances à long terme sont des données qui restent valables durant toute la simulation (identité du personnage par exemple). Les états internes correspondent à l'état mental ou physique du personnage. Ils traduisent, par exemple, sa fatigue ou sa joie. Un plan spécifie une séquence d'actions à accomplir en vue d'atteindre un but. Un plan P_i est défini comme suit : $P_i = (is_i, pc_i, ef_i)$ où :

- is_i est une liste des états internes ;
- pc_i une liste de pré-conditions correspondant à des croyances de l'agent ou à des variables de l'environnement ;
- ef_i une liste d'effets engendrés par un plan : ajout ou suppression de croyances, mouvements de l'humanoïde, changements d'états internes.

Un plan est exécutable si toutes les pré-conditions sont validées et si tous les états internes sont aux niveaux désirés.

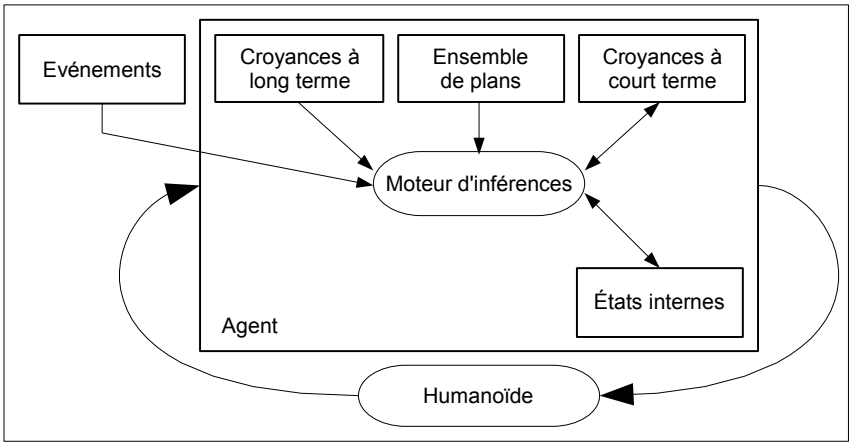

Figure I.11 : *Architecture d'agent BDI pour le contrôle d'un personnage [Caicédo et al., 2000].*

Cette solution est efficace quand l'agent a besoin de raisonner et/ou de communiquer pour arriver à ses objectifs. En revanche, il faut veiller à constituer un ensemble consistant de règles et essayer de limiter les possibilités (rapidité et terminaison des programmes).

Dans le cadre de la navigation, il convient de citer les travaux de [Pettre, 2002] sur la planification de mouvements pour acteurs virtuels. Les résultats présentés bien que spectaculaires sont obtenus dans des environnements statiques avec un seul acteur. Dans le cadre d'environnements dynamiques et continus l'approche réactive propose une alternative intéressante.

Approche réactive

Selon [Duthen, 2002], l'approche réactive (« basée sur l'émergence ») cherche à mieux lier les entités élémentaires du système avec son objectif global ou l'ensemble des contraintes à satisfaire. La modélisation des entités et de l'environnement ainsi que les mécanismes d'auto-organisation ou d'apprentissage doivent permettre aux entités de résoudre le problème.

Le modèle « Boid » [Reynolds 1987;99] est un modèle d'animation très connu et largement utilisé pour les effets spéciaux des films. Ce modèle permet d'animer des groupes de personnages. Chaque personnage dispose d'un modèle décisionnel simple, un comportement cohérent émerge au niveau du groupe (cf. figure I.12). De même [Helbling et al., 2000] ou [Hostetler et al., 2002] ont utilisé des modèles décisionnels réactifs afin de modéliser le comportement de piétons.

Figure I.12 : *Nuée utilisant des modèles décisionnels réactifs [Reynolds, 1987].*

Pour terminer ce rapide tour d'horizon des modèles décisionnels d'acteur virtuel il convient d'aborder les approches hybrides. Comme nous l'avons précisé dans la présentation des SMA, il n'existe plus de clivage réactif / cognitif et chaque modèle adopte des méthodes issues des deux techniques. Les travaux de [Lamarche, 2003] synthétisent des idées issues de ces deux approches. En effet, ses travaux constituent une synthèse et une amélioration de nombreuses thèses de l'équipe SIAMES de L'INRIA (dont [Moreau, 1998], [Thomas, 1999] et [Devillers, 2001]), ses personnages planifient leurs déplacements, utilisent des méthodes réactives et un algorithme spécifiques pour adapter leur trajectoire, ils sont capables de suivre un scénario, de coordonner leurs mouvements ...

3. Conclusion

Lors de l'introduction générale, nous avons présenté le contexte de travail à savoir la réalisation de piétons virtuels destinés à un simulateur éducatif. Ces piétons virtuels ont un rôle important dans ce contexte mais les difficultés liées à leur réalisation sont nombreuses [Thalmann, 2003]. La présence, dans un même environnement, de piétons autonomes interagissant entre eux et avec un utilisateur nous ont conduit aux systèmes multi-agents. En effet, les SMA traitent de ce type de problème indépendamment de tout contexte applicatif.

Dans ce chapitre introductif, nous avons donc présenté les domaines des systèmes multi-agents et des acteurs virtuels. Après avoir introduit ce domaine, nous avons présenté les cinq facettes des SMA (Agent, Environnement, Interaction, Organisation et Utilisateur) [Demazeau, 1995].

La création de personnages virtuels est un thème de recherche appartenant au domaine de l'animation comportementale, nous avons présenté ce domaine en nous appuyant sur les facettes présentées dans le cadre des SMA.

Les acteurs virtuels évoluent dans des environnements dynamiques et continus. Les recherches sur ces environnements ont permis d'améliorer leur modélisation et de simplifier les modules de perception et décision des acteurs virtuels. Les formes d'interactions possibles entre acteurs sont très nombreuses. Néanmoins, les exemples relatifs à l'organisation de sociétés d'acteurs virtuels sont assez rares. Dans notre application, les agents sont d'ailleurs le plus souvent socialement autonomes. Les problèmes liés aux interactions entre agents doivent donc être résolus par les modèles décisionnels des acteurs eux-mêmes. Or la majorité des modèles décisionnels est basée sur une approche impérative [Duthen, 2002] qui oblige le concepteur à décrire le comportement des acteurs face à toutes les alternatives. Les approches cognitives apportent des améliorations. Néanmoins elles peuvent difficilement prendre en considération l'aspect continu des environnements virtuels réalistes. Dans ces conditions, l'autonomie des personnages est donc relativement restreinte. L'approche réactive a déjà été utilisée avec succès dans plusieurs applications [Reynolds, 2000] [Helbling et al., 2000] car elle est particulièrement adaptée aux environnements dynamiques, continus et inaccessibles.

Partant de ces constations, nous consacrerons notre second chapitre au problème de la sélection d'actions. Ce problème reste l'un des enjeux majeurs pour les systèmes multi-agents et les acteurs virtuels.

Chapitre II : Sélection distribuée d'actions

Dans le premier chapitre nous avons présenté les SMA et le domaine de l'animation comportementale. Notre objectif est de donner de l'autonomie aux acteurs virtuels que sont les piétons virtuels afin qu'ils puissent gérer les interactions entre eux, avec leur environnement et avec l'utilisateur. Ces agents ne pouvant s'appuyer sur des organisations pour résoudre leurs problèmes d'interactions, doivent disposer de modèles décisionnels performants.

Nous avons cité quelques exemples de modèles décisionnels dans le premier chapitre et nous nous proposons d'approfondir leur étude dans ce second chapitre. En particulier, la navigation autonome est un problème de sélection d'actions qu'il convient de traiter.

Ce second chapitre comporte quatre parties. La première partie est une présentation de la sélection d'actions et plus particulièrement de l'approche comportementale. Nous rappelons notamment que les modèles orientés comportement peuvent être scindés en deux principales catégories : les modèles utilisant l'arbitrage et les modèles qui utilisent la fusion d'actions. La seconde partie concerne les architectures les plus significatives utilisant l'arbitrage. La troisième partie aborde les architectures à base de fusion d'actions et notamment celles qui utilisent le vote. La quatrième partie nous permet de synthétiser l'ensemble des idées abordées dans le chapitre. Nous montrons les raisons qui nous conduisent à suivre une approche à base de vote ainsi que les différences entre ces architectures. La présentation des inconvénients de cette méthode nous conduit finalement à envisager différents objectifs d'amélioration.

1. Sélection d'actions

Dans cette première partie nous définirons le problème de la sélection d'actions et les caractéristiques d'un modèle décisionnel. Nous approfondirons l'approche orientée comportement après avoir rappelé les limites de l'approche cognitive.

1.1 Définition

La sélection d'actions est le problème majeur auquel chaque agent est confronté : comment choisir l'action la plus pertinente face à une situation donnée.

> *"How can an agent select "the most appropriate" or "the most relevant" next action to take at a particular moment, when facing a particular situation?" [Maes, 1989]*

D'une manière générale, notons les composants nécessaires à la sélection d'actions (illustrée par la figure II.1) :

- $\mathbf{qe_t}$ la perception du monde à l'instant t,
- **ci** les caractéristiques constantes propres à l'agent,
- $\mathbf{qi_t}$ le vecteur de configuration à l'instant t,
- $\mathbf{v_t}$ le vecteur de commande à l'instant t,
- **f** la fonction d'évolution qui permet de calculer le vecteur de configuration au pas suivant $\mathbf{qi_{t+1} = f(qi_t, ci, qe_t, v_t)}$.

Le mécanisme de sélection d'actions est alors une fonction **g** qui établit le vecteur de commande pour le pas suivant ; ce que l'on note : $\mathbf{v_{t+1} = g(qi_t, ci, qe_t, v_t)}$.

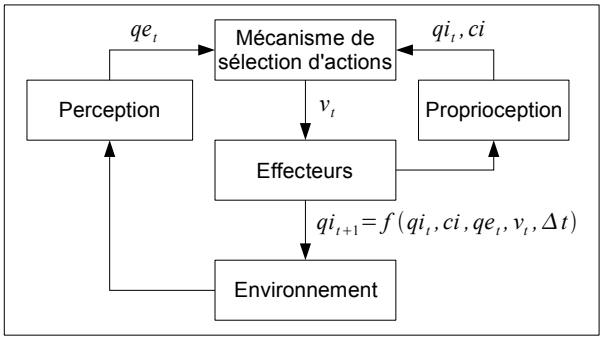

Figure II.1 : *Sélection d'actions dans le cycle d'un agent.*

Par exemple, dans le cadre de déplacements autonomes :

- $\mathbf{qe_t}$ est une liste de perceptions : obstacles, voies, etc.
- Les composantes de **ci** représentent les dimensions du mobile, sa masse, etc.
- Les composantes de $\mathbf{qi_t}$ sont la position et l'orientation du mobile.
- Les composantes de $\mathbf{v_t}$ sont les consignes de vitesse, de direction, etc.
- La fonction d'évolution **f** traduit la dynamique du mobile ; elle permet de calculer ses nouvelles positions et orientations.
- Le mécanisme de sélection d'actions **g** calcule les consignes $\mathbf{v_t}$ à appliquer au mobile.

Un mécanisme de sélection d'actions prend donc en compte un nombre important de paramètres. La sélection d'actions peut être vue comme un problème d'optimisation, mais s'en distingue par la prise en compte de la dynamique de l'environnement. Par ailleurs, le problème est d'autant plus difficile à modéliser que cet environnement est imprévisible et continu. Dans ce contexte, il n'est pas possible de faire des choix optimaux, mais ceux-ci doivent être « suffisamment bons ». En général, il est difficile de qualifier ce qu'est un « bon » modèle décisionnel ; [Pirjanian, 1999] rappelle que peu d'approches définissent une mesure pour évaluer la sélection d'actions.

Nous allons néanmoins essayer de caractériser les modèles décisionnels dans le paragraphe suivant.

1.2 Caractéristiques des modèles décisionnels

Bien qu'il semble impossible de quantifier la pertinence d'un modèle décisionnel, un premier critère général de classification est le type d'actions qu'il est capable de sélectionner. Comme l'ont fait [Maes, 1989] et [Tyrrell, 1993], il est également possible de donner quelques recommandations au concepteur.

1.2.1. Modélisation de l'action

Avant d'être sélectionnée par un modèle décisionnel, une action doit être préalablement modélisée. Il est important de ne pas confondre l'action et son effet sur l'environnement (Voir chapitre 4 de [Ferber, 1995] pour les effets sur l'environnement). Selon (entre autres) [Girard, 2003] un modèle décisionnel sélectionne les actions dans un domaine (ou espace d'actions) pouvant être discret ou continu.

Pour que le modèle décisionnel manipule un espace d'actions discret, le concepteur du modèle constitue le répertoire des actions pertinentes. En fonction de leur granularité, les actions discrètes sont classées en deux catégories [Girard, 2003] : les actions élémentaires ou intégrées. Les actions élémentaires sont simples et directement applicables. Il s'agit, par exemple, d'actions comme « avancer d'un pas dans une direction donnée ». Les actions intégrées sont plus complexes et nécessitent l'exécution d'un algorithme quelconque. Il peut s'agir d'actions comme « aller chercher de la nourriture » par exemple. Il peut y avoir plusieurs alternatives à la réalisation d'une action intégrée, dans ce cas le problème de sélection d'actions se pose à nouveau.

Les actions peuvent également être sélectionnées dans un espace continu. Le modèle décisionnel sélectionne alors des variables numériques qui seront directement appliquées aux effecteurs de l'agent. Un avantage est qu'il n'est plus nécessaire de construire un répertoire d'actions possibles mais l'ensemble des possibilités est infini.

Un modèle décisionnel peut traiter des actions continues, discrètes, élémentaires ou intégrées mais cela ne constitue qu'une caractéristique (certes importante) parmi d'autres.

1.2.2. Qualités d'un modèle décisionnel

[Tyrell 1993] a utilisé des « animats »[8], pour comparer différents modèles décisionnels. Ceci lui a permis de dégager, en s'inspirant de [Maes 1989], les qualités que devrait posséder un modèle décisionnel. Ces recommandations concernent à la fois la stratégie qui permet d'effectuer le choix, mais également les qualités du modèle en lui même.

[Maes 1989] et [Tyrell 1993] postulent ainsi que le choix devrait être :

1) **Dirigé par des buts.** Le choix doit naturellement contribuer à la réalisation d'un ou de plusieurs objectifs de l'agent ou du SMA. Un but correspond à un état de l'environnent ou de l'agent. On distingue deux types de buts (voir par exemple [Dorer, 1999]) : les buts de réalisation et ceux de maintenance.

- Si un but de réalisation (*achievement goal*) est abandonné avant sa réalisation, le bénéfice des actions menées pour l'atteindre est totalement perdu.
- Contrairement aux buts de réalisation, un but de maintenance (*maintenance goal*) ne doit pas nécessairement être pleinement réalisé pour être bénéfique. Par exemple, si un des buts d'un robot est de recharger ses batteries, le but peut être abandonné avant que les batteries ne soit totalement pleines.

Un bon modèle décisionnel devrait pouvoir tenir compte des particularités de chaque type de but.

2) **Situé.** Le modèle décisionnel de l'agent est utilisé dans un environnement. Le choix doit être pertinent par rapport à la situation présente. Le mécanisme de raisonnement est lié avec le monde physique ou simulé.

3) **Anticipé.** Le modèle décisionnel doit prévoir l'effet de ses actions, éventuellement l'évolution de l'environnement pour éviter les situations hasardeuses.

4) **Persistant.** Pour être efficace et atteindre ses objectifs un agent doit persister dans ses choix.

5) **Opportuniste.** Un agent ne doit pas hésiter à rompre la persistance pour réaliser un objectif secondaire facilement atteignable.

6) **Un bon compromis.** Un agent doit favoriser les actions qui lui permettent de satisfaire un maximum de ses objectifs en respectant ses contraintes.

Pour [Maes, 1989] et [Tyrrell, 1993] le modèle doit également posséder quelques qualités fonctionnelles, c'est à dire être :

7) **Réutilisable.** Le modèle doit être adaptable à différents niveaux d'abstraction : choix stratégiques, tactiques et opérationnels. Néanmoins, il semble assez utopique de prétendre concevoir des modèles adaptables à différents problèmes. [Girard, 2003] prend par exemple le cas des animats et rappelle que peu de modèles sont capables de traiter conjointement les choix motivationnels et la navigation.

8 L'approche animat (contraction d'animal et d'artefact) se développe depuis une dizaine d'années. Elle se donne pour objectif la compréhension des comportements adaptatifs basiques comme la locomotion, la navigation, la sélection d'actions ou la coordination d'actions collectives. Selon la vision "animat", ces comportements adaptatifs basiques sont les primitives de processus complexes dont l'étude est une étape préliminaire à la compréhension de la cognition humaine [Guillot, 1999].

8) **Rapide.** Une décision tardive est souvent inutile. Le modèle décisionnel doit fonctionner rapidement pour prendre en compte la dynamique de l'environnement.

9) **Robuste.** Le modèle décisionnel est qualifié de robuste s'il résiste au fonctionnement dégradé de l'une de ses parties. Cette qualité semble difficilement atteignable, l'utilisation de la redondance est une façon d'y prétendre.

Les modèles présentés par la suite seront étudiés par rapport à ces critères. Il faut noter que ces critères sont nombreux, parfois vagues et souvent contradictoires (persistant et opportuniste, par exemple).

Lorsque l'on conçoit un agent, il faut veiller à ce que le modèle décisionnel s'intègre parfaitement avec les autres fonctions principales que sont la perception et l'action. C'est le rôle de l'architecture de l'agent. La conception de cette architecture est importante, non seulement parce qu'elle favorisera ou non le développement et la maintenance de l'agent, mais parce qu'elle a des répercussions importantes sur la manière dont les problèmes sont abordés et résolus. Comme le souligne [Dalgalarrondo, 2001], la conception d'une architecture est particulièrement complexe, puisqu'elle demande une connaissance de tout ce qui devra être intégré grâce à cette architecture. De plus, chaque fonction influe sur le reste du système. Par exemple, la manière dont l'environnement est perçu influe sur la manière de raisonner et inversement.

Lors de la conception d'une architecture agent, il est possible d'adopter une démarche descendante ou ascendante. Cela conduira, respectivement à adopter une architecture fonctionnelle ou comportementale.

1.3 Approche classique cognitive

Le principe de l'approche classique est de concevoir l'architecture du modèle décisionnel suivant une démarche descendante. La figure II.2 illustre le fonctionnement d'une architecture fonctionnelle : chaque module réalise une fonction dont le résultat est utilisé par le module suivant. L'exécution des modules est donc séquentielle. Sur cette figure, les données brutes issues des capteurs sont d'abord traitées et fusionnées pour construire un modèle du monde. Ce modèle est exploité par le module de planification pour réaliser un plan permettant d'atteindre les objectifs. Chaque étape de ce plan est ensuite exécutée. A chaque étape du plan, les consignes sont transmises aux contrôleurs qui pilotent les actionneurs de l'agent.

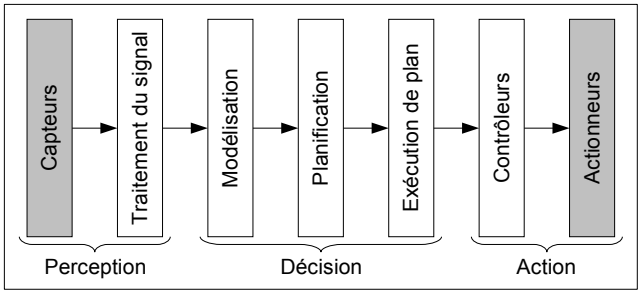

Figure II.2 : *Architecture fonctionnelle résultant de l'approche descendante.*

Cette approche souffre de trois principales limitations : le problème de l'ancrage des symboles, le problème du cadre ainsi que le problème de fragilité.

Le problème de l'ancrage des symboles (Symbol grounding problem) [Harnad, 1990] est lié au fait que les mécanismes de raisonnements manipulent des symboles. Par conséquent, l'interaction entre le monde réel ou simulé n'est possible qu'à condition d'établir une

correspondance entre les symboles et les objets réels. Cela est impossible dans des environnements continus.

Les problèmes de cadre (Frame problem) et de fragilité (Brittleness problem) ont été mis en avant dans [Pylyshyn, 1987]. Ils sont directement liés au problème de l'ancrage des symboles. Premièrement, on ne peut pas représenter un environnement complexe avec peu de symboles. Deuxièmement, il y a risque d'explosion combinatoire si on essaie d'évaluer toutes les possibilités. Troisièmement, la vision limitée d'un agent ne lui permet pas de construire un modèle complet du monde. Enfin, la fragilité du modèle décisionnel sera criante dès lors qu'un agent se trouve face à une situation imprévue.

En outre, de par son fonctionnement séquentiel, ce type d'architecture risque d'être exécuté trop lentement pour prendre en compte la dynamique de certains environnements. De plus, les erreurs se propagent au travers de chaque module.

Même si l'approche classique possède ces quelques défauts, il faut tout de même souligner que des compétences cognitives sont souvent nécessaires à un agent autonome. Les approches basées sur les comportements sont une alternative à l'approche classique. Elles ont pour but principal de mieux lier les agents à leur environnement (critère n°2 : situé). En particulier, l'approche comportementale oppose l'incarnation physique dans l'environnement (physical grounding hypothesis) à l'hypothèse de l'ancrage des symboles. Elle possède d'autres particularités que nous allons présenter.

1.4 Approche comportementale réactive

1.4.1. Origines

L'approche classique et l'architecture fonctionnelle associée possèdent quelques défauts particulièrement pénalisants dans le cadre des agents situés. L'approche comportementale présente une alternative intéressante. Son origine (la plus ancienne) est le courant psychologique du même nom (aussi appelé « behaviorisme »), né dans les années 20. Ce courant psychologique est particulièrement important puisqu'il est le premier à avoir abordé ce thème de manière scientifique [Kennedy et al., 2001]. En effet, il est fondé sur l'étude des seules données observables : les comportements. L'approche comportementale tend à représenter l'intelligence par des ensembles de comportements, issus de conditionnements, et déclenchés par des stimuli indépendants d'états mentaux internes [Dalgalarondo, 2001].

La psychologie cognitive a suivi l'approche comportementale. Elle date des années 50 et a largement inspiré l'IA puisque la métaphore qui la définit présente l'esprit comme un programme exécuté par une machine qui ne serait rien de moins que le cerveau humain [Kennedy et al., 2001].

L'approche comportementale a été reprise à la suite du relatif échec de l'IA en robotique. Elle a, en effet, inspiré la robotique de la fin des années 80 ([Brooks, 1986], [Arkin, 1989] [Rosenblatt et al., 1989]) ainsi que des théories sur l'intelligence [Minsky, 1988]. Les recherches menées dans ce sens ont donc introduit un renouveau dans le domaine en empruntant des idées relativement anciennes. Cela s'est traduit par des travaux sur des robots aux capacités de raisonnement symbolique réduites mais favorisant l'interaction avec le monde réel. C'est ce que Brooks nomme la « physical grounding hypothesis ».

1.4.2. Principe

Cette rapide introduction, a permis de rappeler les origines de l'approche comportementale et ses objectifs. Nous devons à présent présenter les bases de cette théorie. Le postulat fondateur de l'approche comportementale est qu'un modèle décisionnel peut être réparti entre différentes entités plus ou moins indépendantes : les comportements. La figure II.3 illustre la volonté de distribution du modèle décisionnel : la sélection d'actions y est répartie entre différents comportements indépendants.

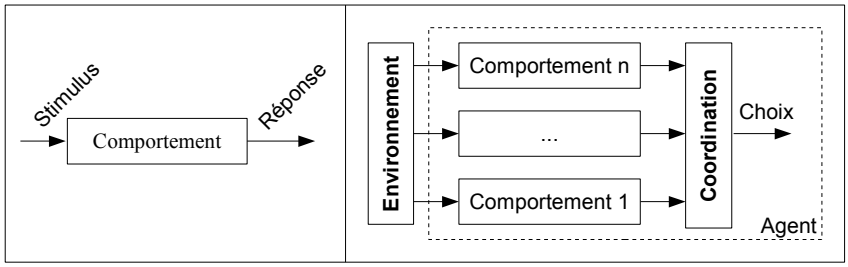

Figure II.3 : *comportement et modèle décisionnel orienté comportement.*

Chaque comportement gère un aspect particulier du problème. Par exemple, pour les robots mobiles de Brooks, un comportement est responsable de l'évitement d'obstacles, un second des déplacements aléatoires et un troisième de l'exploration de l'environnement.

Concrètement chaque comportement :

- possède la connaissance des objectifs et contraintes qui le concernent. En effet, la décomposition du modèle décisionnel en comportements permet de prendre en compte le fait que l'agent possède plusieurs buts. Ces buts peuvent correspondre à l'accomplissement de divers objectifs mais également au respect des contraintes. Cela correspond à la première qualité d'un modèle décisionnel selon [Maes, 1989] et [Tyrrell, 1993].
- est chargé de la perception des stimuli associés. La division des tâches de perception entre les comportements et leur indépendance implique qu'il n'existe plus de mémoire centrale. Cela permet de s'affranchir des problèmes de fusion des perceptions et de gestion de la cohérence. De cette manière, le lien est donc effectivement plus fort avec l'environnement. malgré une possible redondance de certaines tâches de perception.
- peut agir en conséquence en proposant une réponse adaptée. Compte tenu de l'objectif associé à un comportement et à l'état perçu du monde, un comportement propose une réponse.

Puisque chaque comportement propose des options, les architectures comportementales doivent proposer une méthode permettant de gérer les conflits potentiels. C'est le rôle du système de coordination (figure II.3) qui sélectionne les actions en fonction des propositions des comportements. Son fonctionnement sera détaillé par la suite, avant nous approfondirons la notion de « comportement ».

1.4.3. Comportements

Le comportement est un objet abstrait, « brique » de base des architectures comportementales. Les comportements perçoivent les stimuli qui les concernent et proposent des réponses.

Un comportement peut ne pas posséder de but. Dans ce cas, il est dit dirigé par les événements (event driven) [Dalgalarrondo, 2003] (cf. critère n°2). Par exemple, le comportement « éviter les obstacles » de [Brooks, 1986] réagit uniquement à la présence du stimulus « obstacles » et propose de dévier la route du robot. Un comportement dirigé par les événements propose donc le plus souvent des réponses réflexes.

Un comportement peut être dirigé par un ou plusieurs buts (goal driven) (cf. critère n°1, paragraphe 1.2.2). Dans ce cas, ses propositions contribuent à leurs réalisations. Le comportement doit alors mettre en oeuvre un algorithme lui permettant d'adapter ses propositions.

Ainsi s'il est possible de distinguer les comportements selon qu'ils sont dirigés par des buts ou par l'environnement. Il faut également différencier les comportements selon les algorithmes qu'ils mettent en oeuvre.

Les comportements développés sont essentiellement réactifs. Par cela, nous entendons qu'il existe un déterminisme strict entre les stimuli et l'action proposée par le comportement. Un comportement réactif peut donc être décrit par une liste de réactions. Par conséquent, le temps d'exécution d'un comportement est bref (par rapport à un raisonnement cognitif). Ainsi un modèle décisionnel composé de comportements réactifs respecte le critère de rapidité (critère n°8). Il faut cependant ajouter que cela se fait parfois au détriment de la pertinence des choix.

Certaines applications utilisent des comportements cognitifs qui vont permettre de planifier des actions (cf. critère n°3). Par exemple, dans l'architecture de [Rosenblatt, 1996], l'un des comportements est un planificateur qui permet au robot mobile de choisir un itinéraire. L'intérêt d'une architecture comportementale est donc de permettre d'associer différents comportements délibératifs ou réactifs, dirigés par des buts ou par les événements. Le mécanisme de coordination doit permettre d'associer tous ces types de comportements.

1.4.4. Coordination des comportements

Les comportements composant un modèle décisionnel gèrent des aspects spécifiques du problème et sont potentiellement très différents. Il convient donc d'élaborer des méthodes efficaces pour coordonner leurs propositions et ainsi résoudre des problèmes complexes.

Pour [Minsky, 1988] l'esprit humain est lui même un assemblage de nombreuses entités bien trop simples pour être elle-mêmes considérées comme intelligentes : des comportements complexes « émergent » de l'interaction de ces entités. Pour lui, il n'existe pas d'arbitrage central. De nombreuses relations de hiérarchie, de compétition ou de collaboration entre les comportements permettent l'émergence d'un comportement global intelligent. L'approche comportementale est donc souvent liée au concept d'émergence (cf chapitre I). Or le concept d'émergence nous rappelle qu'elle n'est qu'une expression qui dissimule mal notre ignorance des systèmes complexes. [Pirjanian, 1998] rappelle que l'émergence va de pair avec l'imprévisibilité. Par conséquent cette vision de l'émergence ne peut être celle d'un ingénieur soucieux de construire des agents capables de réaliser certaines tâches et répondant à des spécifications. Par conséquent la méthode de coordination de comportements repose donc le plus souvent sur un mécanisme de coordination central, chargé de sélectionner l'action qui répond le mieux aux différentes options.

Un mécanisme de coordination d'actions peut mettre en oeuvre différentes politiques. La figure II.4, extraite de [Pirjanian, 1999] représente une classification de ces politiques. Les deux principales familles sont l'arbitrage et la fusion d'actions. Les méthodes classées dans le

groupe « arbitrage » sélectionnent un ou plusieurs comportements pertinents qui vont prendre le contrôle de l'agent jusqu'au prochain pas de calcul. La fusion de commandes consiste à choisir une action en prenant en compte les différentes propositions des comportements. Les détails relatifs à chacune des sous-familles font l'objet des parties 2 et 3.

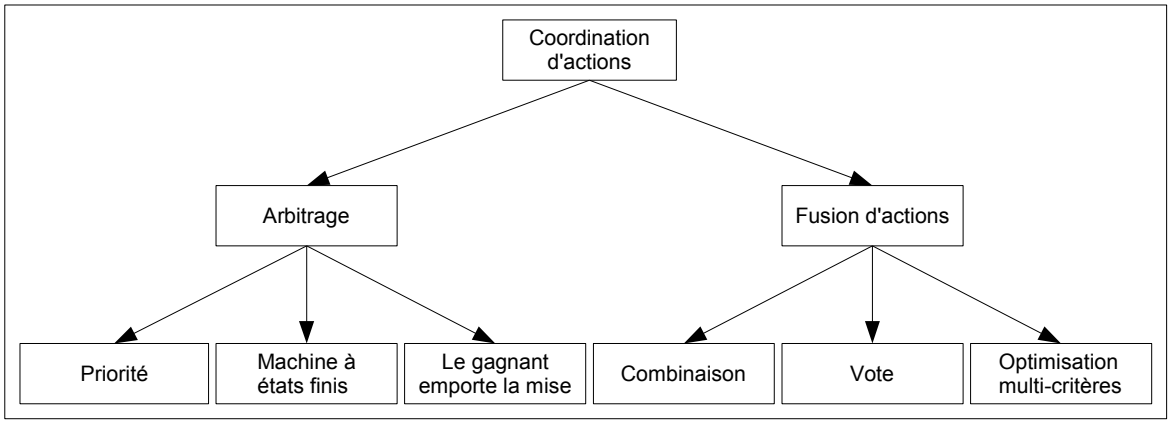

Figure II.4 : *classification des politiques de coordination d'actions [Pirjanian, 1999].*

Le mécanisme de coordination d'actions est le point central des architectures distribuées. Deux principales politiques qui sont *l'arbitrage* et *la fusion d'actions* permettent de coordonner les propositions des comportements. Celles-ci seront détaillées dans la suite de ce chapitre. La figure II.4 illustre leur classification et esquisse ainsi le plan des paragraphes à venir.

2. Coordination de comportements par arbitrage

Des différentes politiques de coordination d'actions, l'arbitrage est la plus simple. Il s'agit de sélectionner un comportement et de lui déléguer le contrôle de l'agent. C'est donc finalement le comportement sélectionné à un pas de calcul qui va, lui seul, choisir les actions réalisées pendant ce pas.

Le problème est donc ramené à la question suivante : « comment sélectionner un comportement et sur quels critères ? » Plusieurs réponses ont été proposées, qu'il est possible de classer en trois stratégies d'arbitrage :

- par priorités,
- par machine à états finis,
- le principe du « le gagnant emporte la mise » (*winner-take-all*).

Les paragraphes suivants présentent ces trois possibilités ainsi que des exemples d'architectures associées.

2.1 Politique d'arbitrage par priorités

Parmi les trois stratégies d'arbitrage, la sélection par priorités est la plus ancienne. Son fonctionnement est le plus simple. Un comportement est actif s'il propose une réponse. Le principe des priorités consiste à sélectionner le comportement actif le plus prioritaire. L'action réalisée par l'agent est donc celle proposée par le comportement actif le plus prioritaire.

C'est sur ce principe que repose l'architecture emblématique de l'approche comportementale : la subsomption [Brooks, 1986] illustrée par la figure II.5. L'architecture globale est divisée en couches. Les plus prioritaires sont les plus hautes et peuvent inhiber ou remplacer les réponses des couches de priorités inférieures (d'où le nom de subsomption). Dans [Brooks, 1986], le niveau zéro est responsable de l'évitement d'obstacles. Le niveau un gère le déplacement aléatoire dans l'environnement tandis que le niveau deux supervise l'exploration de l'environnement. Le niveau trois construit une carte de l'environnement.

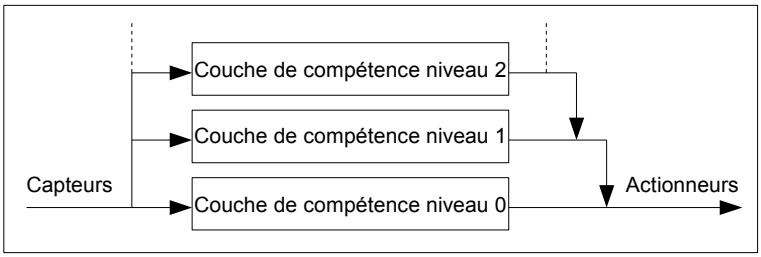

Figure II.5 : *architecture à subsomption [Brooks, 1986].*

Chaque couche est composée de plusieurs comportements. Chaque comportement est décrit par une simple machine d'état qui possède des variables locales, des entrées, des sorties et une entrée de remise à zéro (cf figure II.6). Comme le montre cette figure, les entrées peuvent être subsumées et les sorties inhibées ; les nombres inscrits sur les nœuds correspondent à des temps, ici la sortie est inhibée pendant 3 unités de temps après l'arrivée d'un message sur l'entrée du noeud.

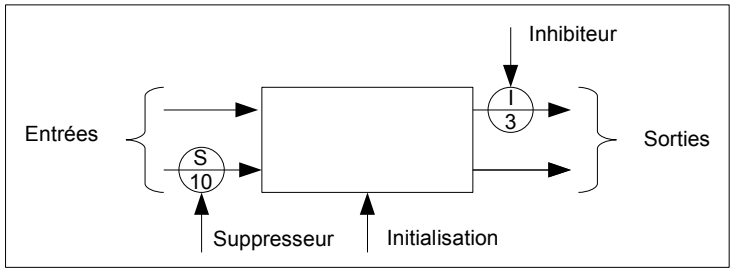

Figure II.6 : *comportements et noeuds de l'architecture [Brooks, 1986].*

En pratique, le réseau n'est pas aussi simple que celui présenté en figure II.5. En particulier, les couches sont fortement liées entre elles : il y a de nombreux échanges entre les comportements de couches connexes. Brooks insiste sur la possibilité de concevoir l'agent de manière incrémentale, c'est à dire en développant un premier niveau puis en ajoutant le second sur le premier sans le modifier et en continuant de cette manière. Cette propriété intéressante est revendiquée par beaucoup d'architectures orientées comportement mais l'expérience montre que l'objectif est rarement atteint. Pour la subsomption [Rosenblatt et al., 1989] souligne qu'un comportement peut subsumer l'entrée d'un autre comportement dont rien ne garantit qu'il ait été prévu ou même testé pour cette nouvelle entrée. Cela montre que la construction incrémentale n'est possible que pour certains cas particuliers.

Rosenblatt déplore également la perte d'informations qui s'opère dans cette architecture car on ne garde aucune trace des propositions subsumées ou inhibées. D'une certaine manière, ce défaut est celui de la stratégie d'arbitrage en général, mais cet effet est aussi accentué par l'utilisation d'un réseau de nœuds complexe.

Le principe de priorité est simple et nous a permis de présenter la première architecture comportementale. La subsomption est historiquement importante. Elle est à la base de l'architecture ALLIANCE [Parker, 1994] ainsi que des robots de [Mataric, 1994]. Cependant, la mise en oeuvre de cette architecture n'est pas aisée car le principe de construction incrémentale n'est pas effectif. De plus il n'est pas toujours possible de mettre des priorités sur les comportements. L'approche à base de machines à états finis qui fait l'objet du paragraphe suivant, présente une possibilité intéressante dans le cas où le concepteur de l'agent souhaite modéliser l'enchaînement temporel des comportements.

2.2 Politique d'arbitrage par machines à états finis

A l'instar de l'arbitrage par priorité, les machines à états finis permettent de sélectionner un des comportements du modèle décisionnel [Blach et al., 1995]. La sélection n'est plus uniquement basée sur des priorités mais fait appel à des machines à états finis qui permettent de modéliser les relations de successions entre les comportements. Lorsqu'un état est actif le comportement associé est sélectionné. Lorsque la transition vers un autre état est vraie l'état actif change et un autre comportement est ainsi sélectionné.

[Bryson, 2002] prône une technique assez proche (les plans réactifs que nous ne détaillerons pas). Elle justifie ce choix par le fait que le concepteur d'un agent connaît souvent l'enchaînement des comportements qu'il souhaite voir réaliser par son agent. L'architecture des schémas [Arkin, 1989] utilise conjointement cette méthode et une stratégie de fusion. (Nous reviendrons sur les schémas dans le paragraphe 3.1).

La simplicité des machines à états finis offre la possibilité de décrire le comportement général de l'agent. Cependant, les difficultés apparaissent si plusieurs états sont actifs en

même temps car plusieurs comportements sont sélectionnés. Pour résoudre ce problème une nouvelle politique de coordination doit alors être mise en oeuvre. Par exemple, dans HPTS (Hierarchical Parallel Transition Systems) [Moreau, 1998][9], le concepteur doit écrire des « fonctions d'intégration » qui permettent de sélectionner une action lorsque plusieurs comportements commandent des actions contradictoires.

Les méthodes par priorités et par machines à états finis sont basées sur des principes très simples. La coordination y est centralisée et le concepteur intervient pour choisir les priorités ou la séquence d'activation des comportements. La méthode suivante se distingue par le caractère distribué de la coordination de comportements.

2.3 Arbitrage par la politique du « gagnant emporte la mise »

Les priorités instaurent une hiérarchie entre les comportements alors que les machines à états finis décrivent leurs successions. La méthode du « gagnant emporte la mise » (Winner take all) est beaucoup moins restrictive. En effet, comme son nom l'indique, il s'agit d'une méthode d'arbitrage mais la stratégie n'est pas explicitement précisée.

La méthode du « gagnant emporte la mise » est inspirée de « la société de l'esprit » [Minsky, 1988] qui imagine l'esprit comme une société bureaucratique de comportements[10] collaborant ou luttant pour le contrôle du corps. Les « réseaux d'activation » (Spreading activation network) [Maes, 1989] constituent une implémentation célèbre de la méthode du « gagnant emporte la mise ». Chaque comportement i constitue un noeud du réseau d'activation et est défini par un tuple (c_i, a_i, d_i, α_i) avec :

- c_i : les conditions selon lesquelles le comportement i peut être activé.
- a_i : la liste des symboles ajoutés dans le modèle du monde si le comportement i est exécuté avec succès.
- d_i : la liste des symboles retirés dans le modèle du monde si le comportement i est exécuté avec succès.
- α_i : le niveau d'activation du comportement i.

Le comportement sélectionné est celui dont l'énergie d'activation α est la plus élevée. L'énergie d'activation est attribuée en fonction de facteurs externes et internes.

Les facteurs externes sont relatifs à l'état du monde et aux buts de l'agent. L'énergie d'un comportement est augmentée s'il est exécutable (si au moins une des conditions c_i est vraie) ou s'il réalise l'un des buts de l'agent (au moins un des ajouts a_i est un but). L'énergie d'un comportement est réduite s'il peut défaire un des buts déjà atteints (au moins une des suppressions d_i correspond à l'un des buts réalisés).

Les facteurs internes sont relatifs aux relations qu'entretiennent les comportements (cf. figure II.7). Les c_i, a_i et d_i permettent de les définir et elles se traduisent par des échanges d'énergie d'activation entre les comportements. Sur la figure, le comportement 2 est le successeur du comportement 1 car des ajouts de 1 sont des conditions de 2. Le comportement 2 transmet donc de l'énergie à 1 pour qu'il soit activé et ainsi rendre une des conditions de 2 vraie. La relation de prédécesseur est définie de manière symétrique. Deux comportements entrent en conflit, si, comme pour les comportements 2 et 3, la condition de l'un a une

9 HPTS n'est pas, à proprement parler, une architecture comportementale, mais utilise des automates pour modéliser des acteurs autonomes. Les problématiques sont donc assez similaires.

10 Minsky emploie le terme agent

intersection avec les suppressions de l'autre. Le comportement 3 limite donc l'énergie d'activation du comportement 2.

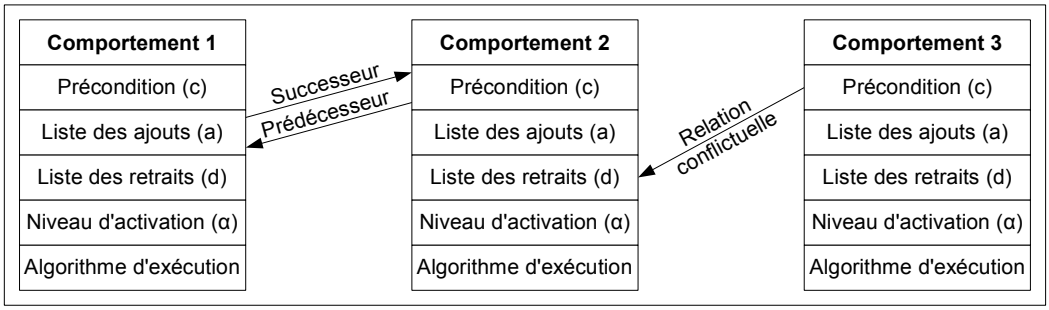

Figure II.7 : *Liens entre comportements dans les réseaux d'activation.*

Les réseaux d'activation ont été repris, entre autres, par [Decugis et al., 1998] et [Dorer, 1999]. Les améliorations modifient l'architecture pour prendre en compte des buts et perceptions continus et également pour proposer des algorithmes d'apprentissage permettant d'optimiser les nombreux paramètres qui définissent les échanges d'énergie.

[Blumberg et al., 1995], repris dans [Isla et al., 2001] apporte une légère modification à la politique du « gagnant emporte la mise » : le comportement sélectionné dirige l'agent mais les autres comportements peuvent influencer la manière dont les actions vont être réalisées. Dans le cadre de l'animation comportementale, le comportement gagnant choisit l'action effectuée mais les autres peuvent affecter l'attitude du personnage ou l'intensité de l'action sélectionnée.

2.4 Conclusion sur la coordination par arbitrage

Les priorités, les machines à états finis ainsi que la méthode du « gagnant emporte la mise » sont les trois principales méthodes de la politique d'arbitrage. L'inconvénient de ces trois méthodes et de la stratégie d'arbitrage en général est que le choix est celui d'un seul comportement, les autres étant totalement ignorés. Par conséquent, cette stratégie enfreint le critère n°6 qui stipule que tout choix doit être un bon compromis.

Cependant cette stratégie est bien adaptée au cas d'une sélection, à un niveau d'abstraction élevé, entre des comportements indépendants et traitant de tâches différentes. Il faut alors souligner que le critère n°7 de réutilisation n'est donc également pas atteint.

La politique d'arbitrage est parfois qualifiée de sélection compétitive. Dans le cas du « gagnant emporte la mise », il existe néanmoins une collaboration entre les comportements. La fusion d'actions, décrite dans le point suivant est qualifiée de collaborative.

3. Politique de fusion d'actions

Contrairement aux techniques précédentes où le système de coordination sélectionnait un comportement ou une action parmi plusieurs possibilités, la fusion d'actions prend en compte les propositions de plusieurs comportements pour essayer d'obtenir une sorte de compromis. Nous retiendrons deux principales stratégies de fusion d'actions : la combinaison d'actions et le vote.

3.1 Architectures à combinaison d'actions

La combinaison d'actions consiste à prendre en compte toutes les recommandations d'actions et à les combiner. Cette stratégie est particulièrement adaptée aux systèmes à base de champs de potentiels. A titre d'exemple, si deux comportements effectuent des propositions différentes, alors l'action retenue est une troisième action qui constitue une sorte de compromis entre les deux propositions.

Parmi les architectures utilisant la combinaison d'actions, nous avons choisi de présenter le modèle « boid » [Reynolds, 1987], les champs de potentiels ainsi que l'architecture à fusion d'actions généralisée [Arnaud, 2000].

3.1.1. Le modèle « boid »

Le modèle « boid » [Reynolds, 1987] est l'un des premiers modèles de simulation centré individu. Il est utilisé pour simuler des bancs de poissons, des formations d'oiseaux ou des mouvements de foule, que ce soit pour le cinéma ou les jeux vidéo [Robert, 2005]. Le comportement du groupe émerge des réactions de chacun de ses membres. Ces entités, appelées « boid » par l'auteur sont capables de se déplacer dans le plan ou l'espace. Chaque *boid* détermine, à chaque pas de calcul, sa direction et sa vitesse. Celles-ci résultent de la combinaison linéaire des accélérations suggérées par plusieurs règles de comportements. Les trois règles de comportements régissant un *boid* dans un groupe sont présentées sur la figure II.8 : la cohésion, l'alignement et la séparation.

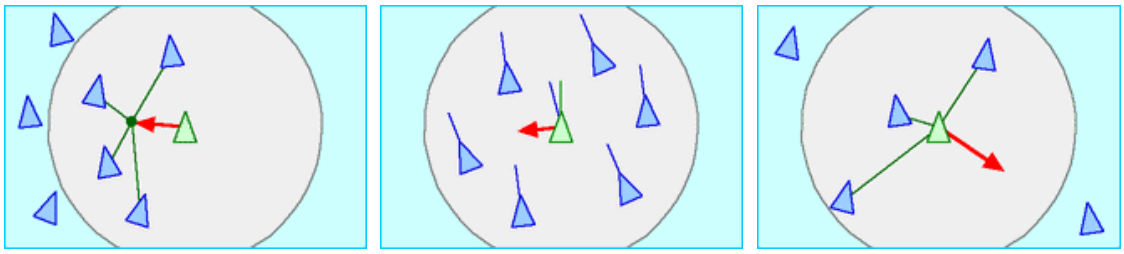

Figure II.8 : l*es trois règles régissant le comportement d'un « boid », de gauche à droite : Cohésion, alignement et séparation [Reynolds, 1999].*

Reynolds a proposé d'autres règles de comportements ainsi qu'un mécanisme d'arbitrage (Prioritized acceleration allocation). Le mécanisme d'arbitrage est rarement implémenté. En revanche, le principe des règles de comportement combinées est encore largement d'actualité. En effet, le principe a été retenu par le groupe navigation de « l'International Game Developper Association »[11]. Leur dernier rapport mentionne de nombreuses règles de comportements comme l'évitement d'obstacles, la poursuite et même le suivi d'un champ de potentiels. Nous allons maintenant présenter cette dernière technique.

11 http://www.igda.org/ai/

3.1.2. Modèles à champs de potentiels

Les modèles à champs de potentiels sont une autre stratégie de combinaison d'actions inspirée par des études sur le comportement animal (cf. [Rosenblatt, 1996]). Cette méthode consiste à assimiler l'agent à une particule plongée dans un champ de forces. Comme le montre la figure II.9, la somme des forces répulsives et attractives permet de générer une trajectoire contournant l'obstacle circulaire.

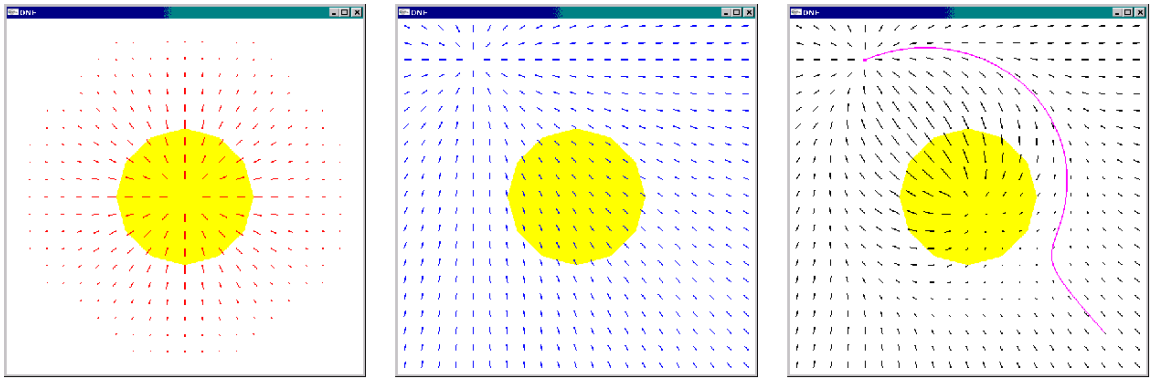

Figure II.9 : *Exemple de champs de potentiels : répulsion, attraction, somme des champs et trajectoire.*

Les comportements utilisés dans ces architectures calculent donc des propositions représentées par des vecteurs. Ces propositions sont fonction de la position de l'agent. La fonction de sélection d'actions calcule donc $\boldsymbol{v_t}$ (l'action sélectionnée) comme une somme de toutes les propositions $\boldsymbol{P_i}$ des comportements i, pondérées par les gains associés $\boldsymbol{G_i}$. Ce que l'on note : $v_t = \sum G_i . P_i$

Les trajectoires sont très souples. En effet, si les fonctions associées aux comportements sont continues alors la décision de l'agent est continue puisque la somme de fonctions continues est une fonction continue. Cependant, dans les applications complexes, les auteurs ajoutent une fonction bruit pour aider l'agent à sortir des minima locaux.

[Zeghal, 1993] a utilisé cette technique dans le cadre de la navigation aérienne. Il a, en particulier, montré que cette technique est applicable aux systèmes multi-agents homogènes.

Les schémas de Arkin [Arkin, 1989] [Blach et al., 1995] fonctionnent également sur ce principe. Les auteurs ont cependant ajouté une dimension supplémentaire. Une bibliothèque de comportements est développée, un séquenceur basé sur une machine à états finis permet de sélectionner les comportements qui vont prendre part à la sélection d'actions.

La méthode a également été appliquée dans des recherches plus récentes. Elle est à la base du modèle satisfaction – altruisme [Simonin, 2001] dédié à la coordination d'agents situés. De même, [Flacher et al., 2003] adjoint aux schémas un algorithme génétique qui, en adaptant les gains des comportements fait émerger une coordination entre des agents réactifs. Notons également que les méthodes de potentiel ont été appliquées aux piétons virtuels [Helbling et al., 2000].

Même si la combinaison linéaire d'actions permet de prendre en compte les différentes propositions cela n'est pas toujours suffisant. En effet, comme l'avait déjà pointé [Reynolds, 1987], il existe des cas pour lesquels un arbitrage est nécessaire. L'architecture à fusion d'actions généralisée propose une réponse à cette constatation en faisant la synthèse d'architectures basées sur la fusion ou l'arbitrage.

3.1.3. Architecture à Fusion d'Actions Généralisée

L'Architecture à Fusion d'Actions Généralisée (AFAG) [Arnaud, 2000] est intéressante car elle constitue une synthèse et une adaptation de différentes architectures : la subsomption [Brooks, 1886], les schémas moteurs [Arkin, 1989 ; 1998] et (dans une moindre mesure) les réseaux d'activation [Maes, 1989]. Cette architecture permet donc de mettre en oeuvre les politiques de fusion ou d'arbitrage.

Dans cette architecture un comportement est caractérisé par :

- **Une activité (*a*)** qui traduit la pertinence d'un comportement en fonction de la présence et de l'importance des stimuli relatifs à ce comportement. Cette activité est un réel dans l'intervalle [0 ; 1]. Un comportement est inactif si son activité est égale à zéro, actif si son activité est égale à 1. Cette activité conditionne le choix d'une action quelle que soit la stratégie adoptée.
- **Un vecteur réel (*R*)** dont *c*haque élément correspond à une consigne sur l'un « des actionneurs » de l'agent, en particulier la vitesse et la direction pour un robot mobile. L'utilisation d'un vecteur et non d'un scalaire prend son sens lorsque le concepteur d'un système basé sur l'AFAG désire mettre en place les méthodes issues de la théorie des schémas (en particulier les techniques à base de champs de forces).

Comme pour la subsomption, le vecteur de consignes appliquées aux actionneurs de l'agent est calculé par un ou plusieurs réseaux de nœuds ; Arnaud en propose quatre types :

- I : Inhibition (architecture de subsomption)
- S : Suppression (architecture de subsomption)
- A : Augmentation (schéma moteur)
- Mx : Maximum (simplification de la sélection d'actions)

Chaque nœud possède deux entrées (directe et transversale) et une unique sortie (partie gauche du tableau II.1). A chaque nœud correspond une fonction de transfert également reportée dans le tableau II.2. Si on utilise des activités booléennes, les nœuds I, S et Mx permettent de mettre en oeuvre une politique d'arbitrage. De même, si les activités sont réelles, les nœuds I, S et A, permettent de mettre en oeuvre une politique de combinaison d'actions. Il faut noter que l'utilisation d'un « nœud A » dans un réseau est hasardeuse puisque son activité en sortie peut être supérieure à 1 (cf. formule 3' dans le tableau ci-dessous).

Tableau II.1 : *Notation et fonctions de transferts des noeuds de l'AFAG*

Entrée transversale (at, Rt) Entrée directe (ad, Rd) Sortie (as, Rs) ?	Inhibition : [1] $R_s = R_d$ [1'] $a_s = a_d (1 - a_t)$	Suppression : [2] $R_s = [a_d (1 - a_t) R_d + a_t R_t] / a_t$ [2'] $a_s = a_d (1 - a_t) + a_t$
	Augmentation : [3] $R_s = (a_d R_d + a_t R_t) / a_s$ [3'] $a_s = a_d + a_t$	Maximum : [4] si ($a_t < a_d$) alors { $R_s = R_d$ et $a_s = a_d$ } Sinon { $R_s = R_t$ et $a_s = a_t$ }

Nous avons implémenté un modèle décisionnel inspiré de [Arnaud, 2000] (cf. [Hanon et al., 2003]). Cela nous a permis de mieux cerner les avantages et lacunes inhérentes à l'AFAG et aux autres modèles fondés sur l'arbitrage ou la combinaison d'actions. Ce modèle décisionnel est simple : la vitesse du mobile est fixe, les changements de direction sont calculés par trois comportements et deux nœuds de coordination (figure II.10).

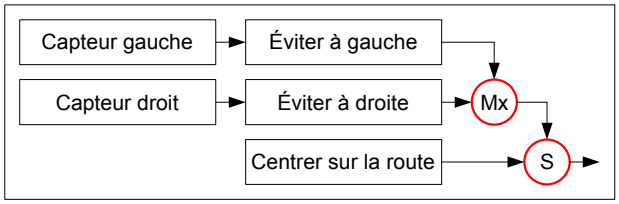

Figure II.10 : *Modèle de navigation basé sur l'AFAG.*

Le comportement « centrer sur la route » (figure I.9) possède une activité *(a)* constante (dans un but de simplification, nous supposons arbitrairement que le mobile doit évoluer au milieu de sa voie). Nous verrons par la suite que la valeur de cette constante a une importance. La réponse *(R)* du comportement dépend de la position de l'agent : il s'agit de suivre la tangente à la voie si l'agent est au centre de la route (avec une distance de tolérance) sinon de dévier d'un angle fixe pour se recentrer.

Les deux comportements d'évitement sont identiques mais utilisés avec des paramètres différents. Contrairement au comportement précédent, la réponse est constante et l'activité variable. Comme le montre la figure II.11, deux capteurs permettent de détecter les obstacles proches de l'agent. L'activité d'un comportement d'évitement est calquée sur celle du capteur qui lui est associé. La réponse d'un comportement d'évitement est une déviation d'un angle fixe dans la direction opposée au capteur associé. Par exemple, sur la figure II.11, l'agent s'écarte vers la droite pour éviter un obstacle à sa gauche. Les capteurs utilisent le principe du « lancer de rayon ». Ils sont caractérisés par leur résolution (nombre entier de rayons), leur portée et leur angle d'ouverture. Ces paramètres sont constants pendant une simulation. Les fonctions d'activation des capteurs retournent un entier entre zéro et un, déterminé en fonction des distances entre l'agent et les points d'intersection des rayons avec les obstacles. Plusieurs fonctions d'activation ont été testées afin, soit d'accorder la même importance à tous les rayons, soit de prendre d'avantage en compte les rayons situés dans l'axe de l'agent.

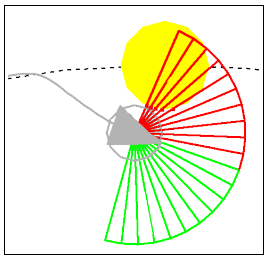

Figure II.11 : *Agent et ses capteurs*
(en gris foncé, le capteur gauche ; en gris clair, le capteur droit).

Pour résumer cette présentation du modèle décisionnel, nous avons synthétisé l'ensemble de ces paramètres dans le tableau II.2. La mise en oeuvre du modèle suppose de définir chaque paramètre. L'ajustement expérimental de ces paramètres est complexe (malgré la simplicité du modèle) car chaque paramètre influe sur le comportement global. Les paragraphes suivants abordent ce problème et présentent les tests effectués.

Tableau II.2 : *Paramètres du modèle décisionnel.*

Paramètre	***Type***	***Objet associé***	***Plage de valeurs***
Ecart latéral : tolérance relative au centre de la voie	Réel (distance)	Comportement *Centrer*	Entre 0 et la moitié de la largeur de la voie
Angle de recentrage	Réel (angle)	Comportement *Centrer*	] 0 ; π/2]
Activité du comportement *centrer* (rapport évitement, centrage)	Réel (rapport)	Comportement *Centrer*	[0 ; 1]
Résolution des capteurs (nombre de rayons)	Entier	Capteurs	[1 ; ∞ [
Portée des capteurs	Réel (distance)	Capteurs	] 0 ; ∞ [
Angle d'ouverture des capteurs	Réel (angle)	Capteurs	[0 ; π]
Fonction d'activation des capteurs	Fonction	Capteurs	Retourne un réel [0 ; 1]

L'environnement dans lequel le modèle décisionnel a été testé est simple. Les agents mobiles simulés évoluent sur une voie circulaire plane encombrée d'obstacles circulaires de rayons variables (figure II.12).

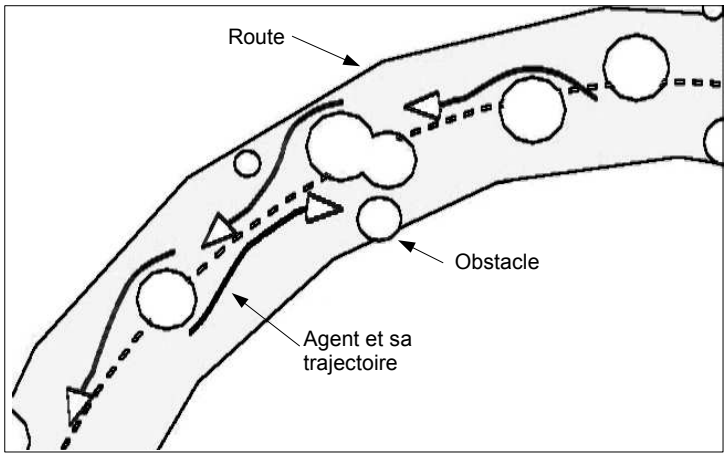

Figure II.12 : *Agents et obstacles sur une partie de la voie circulaire.*

Les trajectoires présentées sur cette figure paraissent cohérentes car elles ne montrent pas les principales lacunes du modèle et, en particulier, les problèmes de persistance des choix (critère n°4) qui se traduisent par des oscillations du mobile. Nous avons effectué une première série de tests simples, significatifs des difficultés de mise au point et permettant de comparer les résultats. Pour cela, nous avons enregistré la trajectoire d'un agent contournant un obstacle, en faisant varier l'activité du comportement *Centrer* (de 0.1 à 0.5) pour changer l'importance relative des comportements. La première série de mesures est présentée à la figure II.13. Les trajectoires les plus proches de l'obstacle correspondent aux valeurs de l'activité de « centrer » les plus élevées. Il y a d'ailleurs collision lorsque celle-ci est trop importante (ici pour a = 0.5).

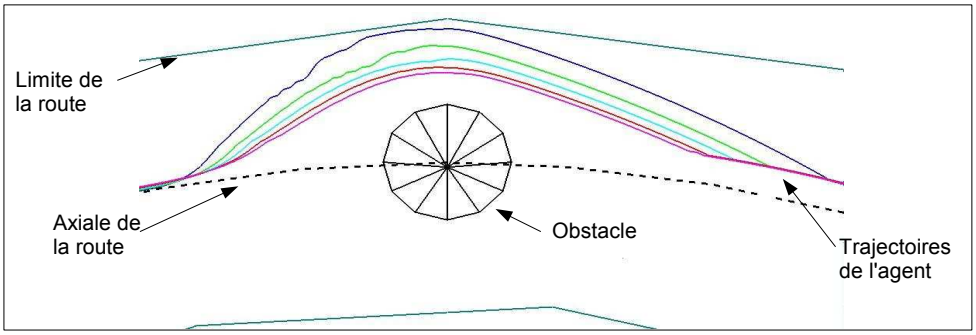

Figure II.13 : *Trajectoires du modèle initial*
(les agents se déplacent de gauche à droite)

Le tableau II.3 est une synthèse de l'ensemble des mesures effectuées. Les cellules sont grisées lorsque l'agent a percuté l'obstacle. « Δ Max » et « Δ Moy » représentent respectivement l'amplitude maximale et moyenne du changement d'angle (en degrés) en sortie du réseau. Ces deux mesures traduisent les changements brusques de direction. Elles permettent de quantifier l'anticipation de l'obstacle (critère n°3). Le nombre de changements de signe de la sortie du réseau traduit l'hésitation. Cette mesure permet de quantifier la persistance du modèle (critère n°4). La première série de mesures correspond au modèle que nous avons présenté. Dans les deux suivantes, les capteurs ont été adaptés : une mémorisation a été ajoutée pour la seconde mesure. Nous avons augmenté l'angle d'ouverture, la résolution et la fonction d'activation des capteurs pour la troisième mesure (la portée des capteurs est restée constante).

Tableau II.3 : *synthèse des mesures effectuées.*

	Pas de mémoire			***Mémoire***			***Mémoire et adaptation des capteurs***		
A centrer	Changements de signe	Δ Max	Δ Moy	Changements de signe	Δ Max	Δ Moy	Changements de signe	Δ Max	Δ Moy
0.1	39	59,2	13,7	18	23,5	3,3	24	14,0	2,1
0.2	36	47,6	8,4	22	21,9	2,6	6	10,4	1,5
0.3	22	28,9	3,6	14	19,7	2,2	4	9,6	1,5
0.4	20	16,9	2,2	16	17,4	2,0	8	10,6	1,5
0.5	24	16,9	2,5	14	16,9	1,5	4	8,8	1,4

Les tableaux II.2 et II.3 nous montrent que beaucoup (par rapport à l'exemple simple) de paramètres interviennent dans l'implémentation d'une architecture AFAG et qu'ils ont un impact important sur la qualité du modèle décisionnel. En effet, dans certains cas, des collisions avec l'obstacle statique sont observées. Ces collisions apparaissent lorsqu'il est accordé trop d'importance au centrage par rapport à l'évitement, le critère n°6 de compromis n'est donc pas atteint. De plus les paramètres ont des répercussions directes sur la persistance (critère n°4) voire sur l'anticipation (critère n°3). Le tableau II.3 montre que les résultats du modèle initial ont été améliorés. Néanmoins, les solutions proposées présentent l'inconvénient d'introduire des paramètres supplémentaires dans le modèle décisionnel. En particulier, la mémorisation demande d'introduire un facteur d'oubli. De même, les fonctions d'activation de capteurs sont paramétrables. Il faut souligner un autre défaut : le modèle décisionnel ne prend pas en compte la vitesse qui influe sur la trajectoire.

Les tests présentés nous ont permis de mieux cerner les difficultés inhérentes à la mise en pratique de l'AFAG ou des architectures du même type. La fusion d'actions par le vote est une alternative à la combinaison d'actions.

3.2 Fusion d'actions par le vote

La combinaison linéaire à la base des modèles précédents pose un problème qui s'avère souvent très pénalisant : cette méthode n'offre aucune garantie sur la validité du résultat. Il est donc possible que l'action résultant de la fusion des préférences ne satisfasse aucune des préférences. Cela se traduit par des collisions dans l'exemple précédent. A l'instar de l'arbitrage, la combinaison de commandes ne respecte donc pas le critère n°6 de compromis.

L'approche dite du vote de préférences distribuées [Rosenblatt et al., 1989] [Sukthankar, 1997] tente de remédier à ce problème. Le principe général de ce type d'architecture est de laisser les comportements évaluer chacune des actions possibles et transmettre ses préférences sous forme de votes à un arbitre central. Lorsque tous les votes ont été transmis, l'arbitre central calcule les scores de chaque action candidate. L'action réalisée par l'agent est donc celle qui a obtenu le meilleur score. Cette démarche présente dans son principe fondateur un avantage certain : l'action sélectionnée est un compromis acceptable pour la majorité des comportements.

Dans la suite de cette section, nous présentons les principales architectures comportementales qui utilisent le vote, c'est à dire :

- DAMN [Rosenblatt et al., 1989], [Rosenblatt, 1996]
- Poly SAPIENT [Sukthankar, 1997],
- Ainsi que les groupes des piétons de [Hostetler et al., 2002]

DAMN est historiquement la première des trois architectures et Poly SAPIENT en est inspirée. Néanmoins, nous présenterons l'architecture de [Sukthankar, 1997] en premier car elle nous semble plus simple à appréhender. En effet, DAMN est davantage appliquée à la robotique et permet d'intégrer beaucoup de comportements très différents. Cela implique quelques adaptations dont les détails rendent sa compréhension moins évidente.

3.2.1. Poly SAPIENT [Sukthankar, 1997]

Le problème soulevé dans la thèse de [Sukthankar, 1997] et dans [Sukthankar et al., 1998] est celui de la simulation de comportements de conducteurs sur autoroute. Dans ce cadre, l'auteur développe et compare deux architectures. La première Mono SAPIENT (Situation Awareness Planer Implementing Effective Navigation in Trafic) est monolithique. La seconde est distribuée : Poly SAPIENT. Le développement de l'architecture distribuée est motivé par la difficulté d'adapter une architecture monolithique à des situations complexes. D'une part, parce que les règles manipulées peuvent interagir de manière imprévisible. D'autre part, parce que l'ajout de nouvelles fonctionnalités dans un unique et conséquent module s'avère difficile. Nous traiterons uniquement de l'architecture distribuée.

Poly SAPIENT est une architecture comportementale qui utilise le vote pour coordonner les préférences. Chaque comportement (Sukthankar utilise le terme « reasoning object ») est responsable d'un aspect très précis de la tâche de conduite, cf. figure II.14.

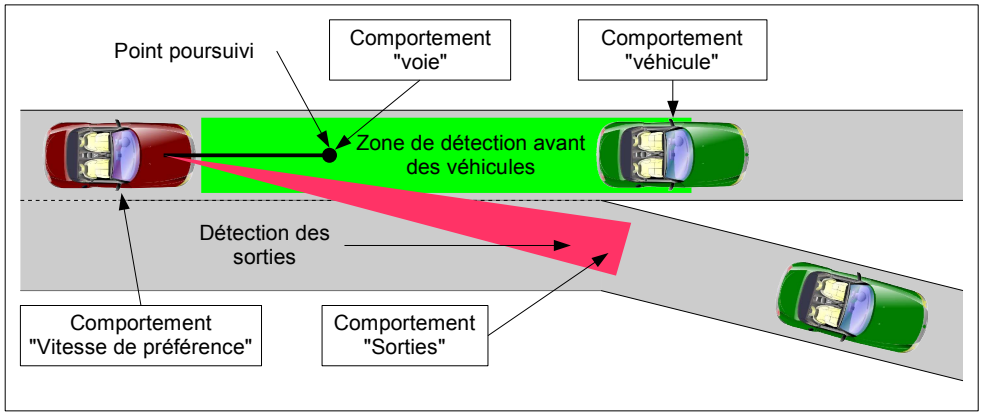

Figure II.14 : *Comportements dans Poly SAPIENT [Sukthankar, 1997].*

Le modèle décisionnel qui regroupe l'ensemble des comportements est présenté sur la figure II.15. Les modules de perceptions (représentés par des ovales) informent les comportements (rectangles) qui transmettent leurs votes à l'arbitre en bas de la figure. Cinq comportements (sur la droite de la figure) prennent en charge l'évitement des voitures. Cela illustre la volonté de distribution fine entre les comportements. En effet, de nombreuses architectures similaires proposent un unique comportement d'évitements d'obstacle. Il faut reconnaître que l'application y est propice car l'organisation de l'environnement la rend évidente. Nous reviendrons sur le comportement *inertie* présent sur la gauche de la figure.

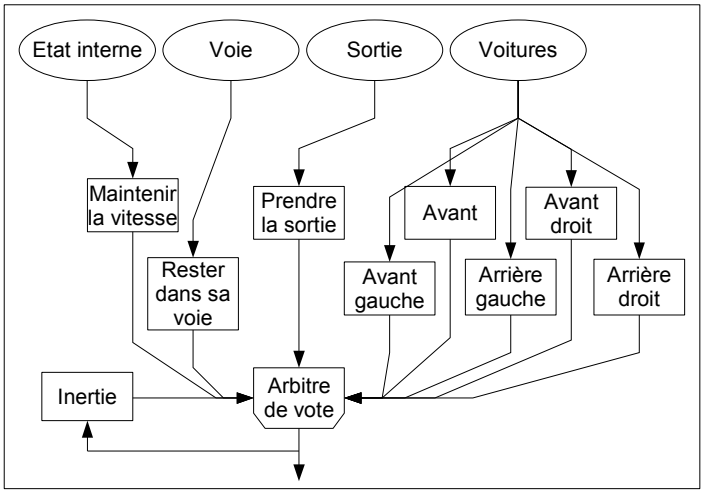

Figure II.15 : *l'architecture Poly SAPIENT [Sukthankar, 1997].*

Chacun des comportements vote pour chaque action de l'espace d'actions. Comme nous allons le voir dans la suite Sukthankar a comparé deux espaces d'actions, « myope » et « global ».

Espace d'actions retenu par Poly SAPIENT

Dans Poly SAPIENT, l'action réalisée par l'agent est choisie parmi un ensemble fini de possibilités (« action space » dans la littérature). Deux espaces d'actions ont été envisagés. Le premier, l'espace d'actions « myope » est défini par rapport à l'agent (le véhicule). Le second, l'espace d'actions global est défini par rapport à l'environnement (ici la route).

L'espace d'actions myope (myopic action space) définit l'ensemble des actions possibles par rapport à l'état courant de l'agent. Dans le cadre de la simulation de véhicules autonomes, chaque action de cet ensemble est un couple de réels qui définit les variations de vitesse et de direction. Sukthankar a retenu cet espace d'actions pour son implémentation et l'a limité à neuf

possibilités. Cela correspond à trois possibilités pour chacun des deux paramètres, vitesse et direction cf. tableau II.4.

Tableau II.4 : *Matrice des possibilités pour le déplacement d'un agent [Sukthankar et al., 1998].*

	Direction		
Vitesse	+ vite à gauche	+ vite tout droit	+ vite à droite
	identique à gauche	identique tout droit	identique à droite
	- vite à gauche	- vite tout droit	- vite à droite

Un second espace d'actions a été envisagé mais n'a pas été retenu : il s'agit de l'espace d'actions global (global action space). Ce dernier décrit l'ensemble des actions possibles par rapport à l'environnement. Dans le cadre de la simulation de véhicules autonomes, la route a été discrétisée en voies et sous-voies. Cette solution n'a pas été retenue car elle soulève quelques problèmes. En particulier, le fait qu'il faille plusieurs pas de temps pour passer d'une voie à une autre, complique inutilement les comportements mis en oeuvre. L'ensemble des actions possibles considérées pour la suite sera donc toujours celui présenté sur la table II.4 qui est défini par rapport à l'agent.

La méthode utilisée dans Poly SAPIENT pour sélectionner les actions est présentée dans le paragraphe suivant.

Sélection d'actions dans Poly SAPIENT

Les comportements sont associés à des objets concrets et l'ensemble des possibilités est un ensemble fini d'actions. Nous définissons la méthode utilisée pour agréger les préférences des comportements.

Une adaptation du vote par estimation moyenne a été utilisée. Chaque comportement attribue une note à chaque option. Les notes sont des réels dans l'intervalle [-1 ; 1]. Par exemple –1 exprime le refus, 0 l'indifférence et 1 l'adhésion totale. Chaque comportement se voit attribuer un coefficient qui traduit son importance. Les notes sont multipliées par ces coefficients avant d'être additionnées pour obtenir une note globale pour chaque option. L'option dont la note est la plus haute est retenue. Sukthankar utilise également la notion du veto qui n'est utilisée dans aucune autre architecture. L'utilisation de veto constitue pourtant un avantage car il permet à un comportement de veiller à ce qu'une action très néfaste (conduisant à une collision ou à la destruction de l'agent) ne puisse être sélectionnée.

Le comportement « inertie » joue un rôle non négligeable dans la sélection d'actions et pourrait être intégré à l'arbitre. Il vote en permanence pour les choix effectués au pas précédent. Cela donne la priorité aux actions en cours et favorise la persistance (critère n°4). L'agent garde ainsi une cohérence dans le déroulement des actions sans nuire à l'indépendance des comportements. L'auteur examine deux possibilités pour réaliser le comportement d'inertie. La première possibilité consiste à superposer le vote de l'instant présent avec le vote du pas précédent. La seconde possibilité est de renforcer uniquement l'action sélectionnée au pas précédent. Sukthankar retient la seconde méthode qui favorise davantage l'action exécutée au pas précédent.

Un algorithme d'apprentissage est également mis en oeuvre pour optimiser les paramètres internes aux comportements ainsi que leurs poids. Poly Sapient est donc une architecture

distribuée utilisant le vote et les veto. La granularité des comportements mis en oeuvre est assez faible mais cela est possible grâce à la structure de l'environnement. Selon Sukthankar, Poly SAPIENT est plus adaptée aux environnements dynamiques que DAMN. En effet, cette architecture propose d'autres utilisations du vote. Celles-ci sont davantage utilisables en environnements statiques, nous aborderons cela dans le paragraphe suivant.

3.2.2. DAMN [Rosenblatt et al., 1989 - 2002]

Présentation

DAMN (Distributed Architecture for Mobile Navigation) [Rosenblatt, 1996], est une architecture orientée comportement, pionnière dans l'utilisation du vote. Les premiers articles [Rosenblatt et al., 1989] la positionnent comme « une alternative à la subsomption ». Le problème de l'arbitrage est mis en avant et l'utilisation du vote est appuyée par deux principaux arguments. D'une part, un comportement doit pouvoir évaluer différentes alternatives. D'autre part, pour certains comportements, il est plus simple de pointer les actions qu'il juge néfastes que de proposer une solution.

L'une des différences majeures de DAMN par rapport à Poly SAPIENT est son utilisation dans le cadre de la robotique qui a des répercussions profondes sur l'architecture. DAMN a été testée sur plusieurs applications robotiques : un véhicule terrestre autonome [Rosenblatt, 1996], des robots caristes autonomes pour installations portuaires et le robot sous-marin Oberon dédié à l'étude des récifs coralliens [Rosenblatt et al., 2002]. DAMN a également été reprise par [Tyrrell, 1993] qui a montré par simulation que ce type d'architecture est la meilleure dans le cadre des animats.

La figure II.16 ci-dessous montre une vue d'ensemble du modèle décisionnel développé pour un véhicule autonome dans [Rosenblatt, 1996]. Contrairement à Poly SAPIENT, les poids associés aux comportements ne sont pas appris mais attribués par l'utilisateur via un gestionnaire de modes. Par ailleurs, un modèle décisionnel DAMN peut utiliser plusieurs arbitres. Ce qui, nous le verrons, introduit de nouvelles difficultés.

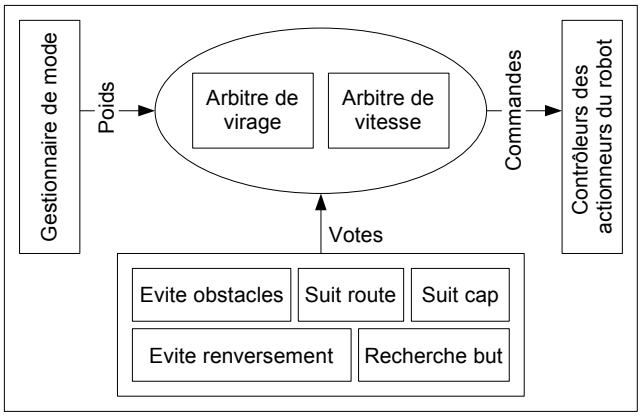

Figure II.16 : *l'architecture DAMN utilisée dans le cadre d'un véhicule autonome [Rosenblatt, 1996].*

Les implémentations de DAMN montrent qu'il est possible d'utiliser des comportements très différents dans une même architecture comportementale. DAMN est fortement inspirée par la subsomption. Or l'une des critiques pointées dans [Rosenblatt et al., 1989] est le fait que les comportements mis en oeuvre par Brooks utilisent uniquement des automates à états finis. Le véhicule autonome est un exemple particulièrement significatif. En effet, les

comportements mis en oeuvre dans son modèle décisionnel sont très hétérogènes, ils ont été développés spécialement ou adaptés d'autres travaux. On peut citer, entre autres :

- Le comportement du suivi de route est basé sur un réseau de neurones. Son adaptation pour DAMN est très simple puisque la couche de sortie du réseau est directement recopiée dans le vecteur des votes.
- Un comportement réalise une interface avec l'utilisateur lui permettant de créer un itinéraire à partir de points de passages.
- Un planificateur permet au véhicule de planifier son chemin.

L'un des avantages de DAMN est donc la possibilité d'utiliser « la meilleure des techniques pour chaque comportement » [Rosenblatt, 1996]. Les implémentations de DAMN montrent également que cette architecture permet effectivement de construire le modèle décisionnel de manière incrémentale. Rappelons que ce but est en réalité difficilement atteint dans les architectures comportementales.

Une des originalités de DAMN réside dans l'existence de différents types d'arbitres.

Arbitres DAMN

Le principe de l'architecture DAMN est simple et général : il s'agit d'une coordination centrale par un arbitre de préférences distribuées. Comme nous allons le voir, différents arbitres ont été conçus et testés sur différentes parties du robot mobile (orientation de la caméra, déplacements du véhicule, etc). L'arbitrage le plus simple est celui par contraintes. L'arbitrage sur l'espace de commandes est similaire à celui de Poly SAPIENT. Les deux arbitres « espace d'action abstrait » et « utilité estimée des actions » sont particulièrement originaux.

Arbitrage par contraintes

Les comportements envoient leurs contraintes à l'arbitre. Il doit donc théoriquement résoudre un problème de satisfactions de contraintes mais dans la pratique, l'implémentation existante est trop simpliste. Comme le montre la figure II.17, chaque comportement envoie à l'arbitre la vitesse maximale acceptable selon le point de vue qu'il considère. L'arbitre sélectionne la vitesse la plus faible qui va donc satisfaire les contraintes exprimées par l'ensemble des comportements. Cet arbitre est peu utilisé.

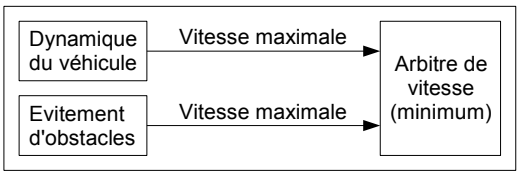

Figure II.17 : *exemple d'utilisation de « l'arbitrage par contraintes » [Rosenblatt, 1996] .*

Arbitrage sur l'espace de commandes.

Cet arbitre fonctionne sur le même principe que l'arbitre de Poly SAPIENT mais diffère dans son implémentation. En particulier parce que les comportements de DAMN sont exécutés sur différentes machines et que leurs propositions ne sont pas nécessairement synchronisées. Rosenblatt [Rosenblatt, 1996] propose de prendre en compte le temps écoulé entre l'envoi du vote et sa prise en compte par l'arbitre. Ce temps mesuré permet de faire

décroître le coefficient associé au vote. Passé un certain temps, le vote n'est plus pris en compte.

Arbitrage sur un espace abstrait d'actions

Il existe donc un problème lié à la non-synchronisation des comportements. L'arbitrage sur l'espace de commande ne prend pas en compte le fait que l'environnement et l'agent évoluent entre le vote et sa prise en compte. Comme les actions sélectionnées sont définies par rapport à la configuration de l'agent (notamment sa position et sa direction), les votes ne sont pas exprimés par rapport au même référentiel. L'arbitre abstrait replace donc les votes dans un même référentiel avant de procéder à la sélection d'actions.

Arbitrage par estimation de l'utilité [Rosenblatt, 2000]

Ce type d'arbitre est assez original puisque les comportements ne votent pas pour des actions mais donnent des informations d'utilités et sur les positions futures de l'agent. Sukthankar [Sukthankar, 1997] note que cela fonctionne bien dans un environnement statique et c'est pour cela qu'il préfère un arbitrage sur l'espace de commandes.

Coordination des arbitres

Pour l'implémentation de ses robots, Rosenblatt utilise différents arbitres. Par exemple la figure II.16 montre deux arbitres, l'un pour la vitesse l'autre pour la direction. Il existe des cas pour lesquels les arbitres peuvent fonctionner de manière complètement indépendante. C'est le cas pour les deux arbitres du robot sous-marin [Rosenblatt et al., 2002]. Généralement, les arbitres doivent échanger des informations et se synchroniser. Par exemple, l'arbitre de vitesse doit prendre en compte l'angle de braquage qui a été sélectionné.

En suivant cette logique de distribution de la sélection d'actions entre plusieurs arbitres on aboutit à l'architecture proposée par [Tyrrell, 1993] inspirée de [Rosenblatt et al., 1989].

3.2.3. Implémentation du vote [Tyrrell, 1993]

L'architecture de [Rosenblatt et al., 1989] est, selon [Tyrrell, 1993], le meilleur des mécanismes de sélection d'actions testé sur ses animats. La figure II.18 ci-dessous représente une partie du modèle décisionnel qu'il a développé pour effectuer sa comparaison (plus précisément, le comportement « avoir de la nourriture »).

Sur cette figure les stimuli internes ou externes sont représentés par des ovales, les comportements et actions élémentaires par des carrés. Les arcs qui les relient sont orientés du haut vers le bas. Cependant, les flèches n'ont pas été reproduites pour plus de clarté. De même, les arcs, allant « d'approcher nourriture » et « chercher nourriture mémorisée » vers les déplacements « normaux » ont été omis mais sont identiques à ceux partant de « approcher nourriture perçue ».

Les liens entre les comportements sont pondérés. L'arc qui relie « Avoir de la nourriture » à « Ne pas gaspiller son énergie » est égal à 0,1. Cette valeur est relativement basse et cela est normal car nous pouvons supposer qu'il existe un comportement « fuir un prédateur » dont les préférences doivent avoir plus d'importance. Les liens partant de « ne pas gaspiller son énergie » vers les déplacements rapides sont égaux à -1.

Contrairement aux autres modèles décisionnels celui-ci est hiérarchisé. La structure hiérarchique est utile lorsque le nombre de comportements est important [Bryson, 2002].

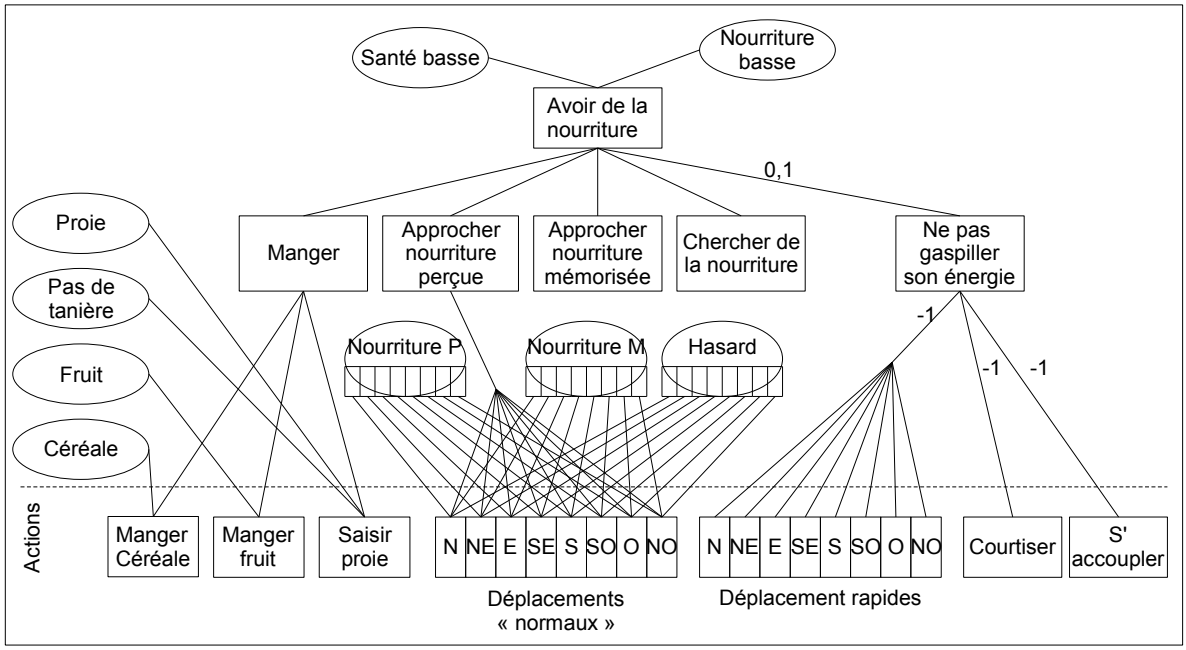

Figure II.18 : *application de l'architecture de [Rosenblatt et al 1989] à un animat [Tyrrell, 1993].*

3.2.4. Groupes des piétons [Hostetler et al., 2002]

Hostetler a développé un modèle décisionnel calqué sur Poly SAPIENT. Ce modèle est utilisé pour simuler des déplacements de groupes de piétons (cf. figure II.19). Par rapport à Poly SAPIENT les comportements possèdent une granularité plus élevée. Les algorithmes de perception sont également différents. Ils divisent l'environnement proche de l'agent en huit zones distinctes. Ce choix s'explique par le fait que le problème des piétons est plus complexe que celui des voitures car il n'existe pas de voies, de sens de circulation, etc.

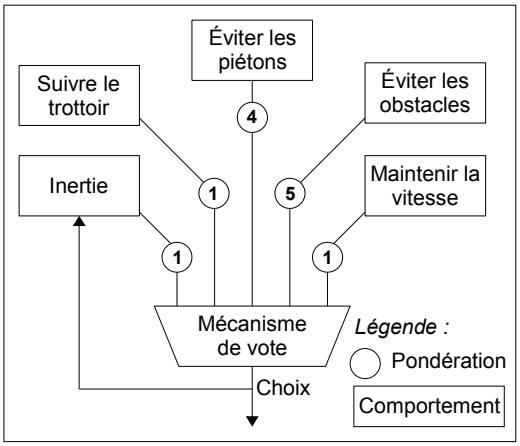

Figure II.19 : *Modèle décisionnel de piéton virtuel [Hostetler, 2003].*

L'une des originalités de l'implémentation de Hostetler réside dans le changement de l'espace d'actions selon la situation. L'espace d'actions utilisé lorsque le piéton évolue librement sur une voie est celui développé par Sukthankar et présenté dans le tableau II.4 (page 52). Lorsqu'un piéton souhaite approcher d'un groupe ou d'un objet d'intérêt, l'espace d'actions change pour devenir tel que présenté sur la figure II.20. En effet, dans ce cas, les comportements ne raisonnent plus sur des variations de vitesse et d'orientation mais directement sur des positions futures. La figure II.20 montre les neuf possibilités retenues pour

les positions futures du personnage ainsi que les trois possibilités pour son orientation. Il y a donc dans ce cas décorrélation de la position et de l'orientation.

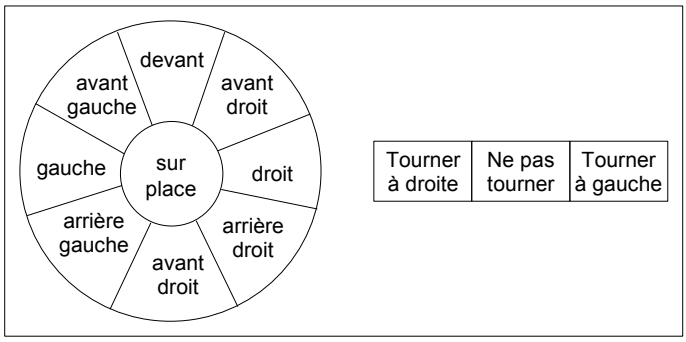

Figure II.20 : *Espace d'actions pour l'approche d'un objet [Hostetler, 2003].*

Le mécanisme de vote est identique à celui de Sukthankar, y compris pour le comportement d'inertie, à la différence près que les veto ne sont pas utilisés et que le choix des poids des comportements est empirique. L'évitement des gros obstacles n'est pas traité par le comportement dédié aux évitements mais par un algorithme qui modifie l'objectif du comportement « suivre le trottoir ». Ce détail montre la difficulté de créer des comportements indépendants qui ne prennent en compte qu'un aspect particulier de la sélection d'actions. Nous reviendrons sur ce problème dans le chapitre IV.

4. Synthèse des architectures orientées comportement

Lors de cette synthèse de la bibliographie, nous comparons les architectures orientées comportement. Nous estimons que la coordination par vote est la méthode de coordination la plus avantageuse. Nous rappellerons donc les différences entre les architectures à base de votes et soulignerons les inconvénients restants. Ces avantages et inconvénients ayant été présentés nous présenterons les objectifs d'amélioration que nous nous fixons.

4.1 Comparaison des stratégies de coordination d'actions

Nous avons présenté des architectures comportementales mettant en oeuvre différentes méthodes de coordination. Pour cette comparaison, nous retiendrons la coordination par arbitrage, par combinaison d'actions et par vote. Comme les architectures que nous comparons sont assez similaires, nous ne retiendrons que trois critères qui sont : le nombre d'alternatives proposées par comportement, la validité de la solution et les domaines d'applications.

Lors de la sélection d'actions, il est important d'évaluer plusieurs alternatives[12]. L'arbitrage ou la combinaison d'actions permettent aux comportements de ne proposer qu'une seule alternative. Or, même si un comportement ne gère qu'un aspect du problème, il doit pouvoir envisager plusieurs alternatives. Avec le vote, l'espace d'actions est établi avant la sélection d'actions, mais les comportements s'expriment sur l'ensemble des propositions. Lorsqu'un comportement vote, il peut donc donner un bonne note à plusieurs options. Grâce à cela, il sera plus facile d'arriver à un compromis (critère n°6) si chaque comportement accepte plusieurs alternatives.

Il convient de discuter de la validité des actions sélectionnées. Dans le cadre des architectures orientées comportement, le nombre de comportements « satisfaits » par l'option sélectionnée nous semble un indicateur de la pertinence du choix. L'arbitrage sélectionne un comportement à qui est confié le contrôle de l'agent ; l'action sélectionnée satisfait dans le pire des cas un seul comportement. La combinaison d'actions semble meilleure sur ce point mais, en réalité, elle n'offre aucune garantie sur la validité de la solution. Dans le pire des cas, une action issue d'une combinaison peut ne satisfaire aucun comportement. Le vote est plus intéressant vis à vis de ce critère puisqu'il est sensé aboutir à un compromis satisfaisant la majorité. La possibilité de déposer des veto est également intéressante puisqu'elle permet de prendre en compte les contraintes imposées par certains comportements. Cette possibilité n'existe dans aucune des autres méthodes.

Toutes les stratégies ne sont pas adaptées aux mêmes problématiques. Par exemple, l'arbitrage convient d'avantage aux choix abstraits et la fusion à des décisions plus concrètes comme la navigation[13]. [Tyrrell, 1993] a montré que le vote est utilisable à tous les niveaux de son modèle décisionnel. D'un point de vue plus général, le vote est une procédure de choix collectif utilisée dans de nombreux contextes.

Les trois points ci-dessus, correspondant aux différences que nous avons essayé de dégager, sont résumés dans le tableau II.5.

12 Il faut cependant éviter d'évaluer trop d'alternatives, l'explosion combinatoire des choix étant courante.
13 D'autant que la fusion n'est possible que sur des données numériques.

Tableau II.5 : *Synthèse et comparaison des divers architectures comportementales.*

	Arbitrage	***Combinaison linéaire***	***Votes distribués***
Références	[Brooks, 1986] [Maes, 1989] [Blumberg et al., 1995]	[Reynolds, 1987] [Arkin, 1989] [Flacher et al., 2003]	[Rosenblatt et al., 1989] [Tyrell, 1993] [Sukthankar, 1997] [Hostetler et al., 2002]
Validité de la solution (en nombres de comportements)	1	0	Majorité
Propositions par comportement	1	1	Espace d'actions
Domaines d'application	Haut niveau d'abstraction	Faible niveau d'abstraction, essentiellement la navigation	Mutiple dont navigation

Le vote constitue la meilleure des stratégies de coordination de comportements. Il permet à chaque comportement d'exprimer ses préférences et ses rejets sur plusieurs options (critère n°2). Ainsi l'option sélectionnée satisfait un maximum de comportements (critère n°6). Le vote est la procédure la mieux adaptée aux décisions collectives. Il est utilisable dans d'autres applications (critère n°7) car il s'applique à tous les niveaux d'abstraction.

Nous avons présenté quatre architectures utilisant le vote [Rosenblatt et al., 1989-02] [Tyrell, 1993] [Sukthankar, 1997] [Hostetler et al., 2002], elles diffèrent sur quelques points soulignés dans le paragraphe suivant.

4.2 Comparaison des architectures à base de vote

Les détails qui diffèrent entre les architectures à base de vote sont : l'espace d'actions, le nombre de propositions par comportement, les priorités, le réglage persistance / opportunisme, la granularité des comportements, leur nombre, la place de la planification, la hiérarchie de l'architecture et enfin le modèle de vote et notamment l'utilisation (ou non) des veto.

Dans la bibliographie toutes les variantes du vote de préférences distribuées utilisent un espace d'action discret. De même, dans la bibliographie l'espace d'action est fixé une fois pour toutes par le concepteur, sauf pour [Hostetler et al., 2002].

Les architectures utilisant le vote font voter les comportements pour chacune des options de l'espace d'action (plus éventuellement pour des sous-comportements dans [Tyrrell, 1993]). Comme nous souhaitons utiliser un espace d'action continu, cela n'est naturellement plus possible.

Toutes les architectures utilisent des coefficients qui pondèrent les votes des comportements. Ces coefficients influent fortement sur le fait que le choix soit un bon compromis. Le problème de choix de ces coefficients est pointé dans [Hostetler et al., 2002]. Poly-Sapient [Sukthankar, 1997] est la seule architecture mettant en oeuvre un algorithme d'apprentissage pour régler les coefficients associés à ces comportements. Cela permet de garantir le fonctionnement de l'architecture mais uniquement pour les situations proches de celles dans lesquelles l'apprentissage a eu lieu.

Pour Poly-Sapient [Sukthankar, 1997] et le modèle d' [Hostetler et al., 2002], un comportement « inertie » permet de régler le problème de persistance des choix (critère n°4 et n°5). Sans ce comportement leurs agents semblent en permanente hésitation.

La granularité des comportements a une influence sur les choix pris par l'architecture. Une granularité faible permet de mieux prendre en considération l'environnement (c'est sur cette

hypothèse qu'est construite l'approche orientée comportement). Pour DAMN [Rosenblatt, 1996] la granularité est très variable. Certains comportements sont purement réactifs et d'autres intègrent de la planification. Pour [Tyrrell, 1993] la granularité est également très variable du fait de la hiérarchie du modèle. [Sukthankar, 1997] réalise une distribution plus fine des tâches entre les comportements parce que son application y est très propice.

Le nombre des comportements est fixe dans toutes les architectures sauf pour DAMN [Rosenblatt, 1996]. En effet un comportement de DAMN peut se voir attribuer temporairement un coefficient nul par le « manager de modes » qui l'exclue ainsi de la sélection d'actions.

Aucune méthode de planification n'est utilisée par [Tyrrell, 1993] et [Sukthanker, 1997] ; pour [Rosenblatt, 1996] la planification se situe à l'intérieur des comportements (granularité) ; dans [Hostetler et al., 2002] il existe une forme de planification à l'extérieur du système comportemental qui règle les buts de certains comportements.

Poly-Sapient [Sukthankar, 1997] est la seule architecture mettant en oeuvre les veto. Ils possèdent une importance capitale pour certaines applications. En effet, un comportement peut être responsable du respect d'une contrainte, le veto lui permet de l'imposer aux autres comportements.

L'utilisation de vote constitue un pas vers la validité des choix retenus par le modèle décisionnel, l'un des principaux inconvénients des autres stratégies (arbitrage et de la combinaison d'actions). L'estimation moyenne est la méthode de vote utilisée par toutes les architectures.

Le tableau II.6 ci-dessous récapitule les différences entre les architectures à base de vote. Dans ce tableau nous avons souligné les solutions qui nous semblent les plus appropriées. Nous avons également essayer d'établir les liens les plus évidents entre les critères spécifiques à ces architectures et les critères généraux de [Maes, 1989] et [Tyrell, 1993].

Tableau II.6 : *Récapitulatif des différences entre les architectures fondées sur le vote. (« D » signifie : rapport direct avec le critère concerné et « I », rapport indirect)*

1) Buts	2) Situé	3) Anticipé	4) Persistant	5) Opportuniste	6) Compromis	7) Réutilisable	8) Rapide	9) Robuste	Caractéristiques de l'architecture	DAMN [Rosenblatt]	[Tyrrell]	Poly SAPIENT [Sukthankar]	[Hostetler]
	D					I			**Espace d'actions**	Discret			
	D					I			**Espace d'actions**	Fixe	Fixe	Fixe	Variable
					D	I			**Veto**	Non	Non	Oui	Non
D	D					I	I		**Nombre de comportements**	variable	fixe	fixe	fixe
			D	D					**Réglage persistance / opportunisme**	Rien	Rien	Comportement d'inertie	Comportement d'inertie
			I	I	D	I			**Priorités entre comportements**	Variable selon le mode	Fixées par le concepteur	Apprentissage	Fixées par le concepteur
I	D				D	I			**Granularité des comportements**	Variable	Selon niveau hiérarchique	Faible	Forte
I		D	I			I	I		**Place de la planification**	Dans un comportement	Aucune	Aucune	En dehors du système comportemental
					D	I	I	I	**Méthode de vote**	Estimation moyenne			

Même si le vote nous semble un excellent moyen de coordonner des comportements, il convient néanmoins de pointer ses quelques défauts.

4.3 Inconvénients des architectures à base de vote

Limiter, comme dans la bibliographie, les possibilités à neuf actions est beaucoup trop restrictif et arbitraire car il y en a souvent beaucoup plus. De plus il faut choisir quelles actions seront pertinentes. Dans le cadre de déplacements, la vitesse et le cap d'un agent étant continus il existe une infinité de choix. Il faut alors choisir un pas d'accélération et de rotation. Si un pas est trop petit, le mobile n'est pas réactif, l'agent ne réagit pas assez vite. Si le pas est trop grand alors les variations de cap et de vitesse seront très fortes. De plus un phénomène permanent d'hésitation risque de se produire puisqu'il est impossible de négocier correctement une trajectoire courbe en tournant avec des angles fixes. Le critère n°4 de persistance n'est donc pas respecté dans ce cas.

La pondération des comportements soulève aussi quelques problèmes. La figure II.19 (page 56) montre, par exemple, un rapport de cinq entre les poids de certains comportements. L'écart est si important que les décisions de quelques comportements prédominent largement. Dans ces conditions le choix ne peut être considéré comme un « bon » compromis (critère n°6). Le vote perd donc son avantage sur les autres méthodes d'arbitrage ou de fusion. Selon les auteurs, la prédominance naturelle de certains comportements suffit à justifier cette différence d'influence. Cet argument est probablement justifiable dans certains cas, mais difficilement dans l'application présentée. Selon la figure II.19, éviter les piétons est plus important que de suivre le trottoir ou d'éviter un obstacle. Dans ce cas l'évitement d'un autre piéton peut provoquer une sortie de route ou une collision avec un obstacle. Par ailleurs le

choix des pondérations peut s'avérer délicat. Hostetler attribue expérimentalement les pondérations. Un algorithme génétique est utilisé par Sukthankar pour les optimiser. Dans les deux cas, une variation des situations auxquelles sont soumis les agents impose de modifier les pondérations. L'intérêt de l'algorithme génétique est ainsi limité par son fonctionnement nécessairement hors ligne.

La granularité des agents réalisés est également parfois discutable. La granularité des agents mis en œuvre dans les études citées précédemment est souvent très grande ; les problèmes dont leurs comportements ont la charge sont très complexes. Or, les architectures comportementales s'inscrivent dans une volonté de distribution et de réactivité qui ne peut être compatible avec des granularités élevées. Dans cette optique, il faudrait proscrire l'usage de comportements tels qu' « Eviter les piétons » (figure II.19). Par ailleurs, il faut rappeler que pour simplifier ce comportement (*a priori* complexe) l'agent présenté dans [Hostetler et al., 2002] ne prend en compte que l'obstacle le plus proche. Il faut cependant noter que la granularité des comportements développés dans [Sukthankar, 1997] est beaucoup plus faible mais que cela est rendu possible par la configuration de l'environnement spécifique à l'application.

Nous avons déjà évoqué l'espace d'action et la pondération. La procédure de vote mérite également d'être clarifiée. En premier lieu, le terme « vote » est employé dans la bibliographie, « agrégation de préférences » serait plus indiqué. En effet, il s'agit de décider quelle option est « collectivement » la meilleure en se basant sur des préférences individuelles. La procédure employée dans la bibliographie est toujours celle du vote par estimation moyenne. Or, il existe une multitude de procédures de vote et cette méthode est très discutable. En particulier, il faut que l'attribution des notes soit parfaitement équitable. Le concepteur doit respecter une échelle de notation et les pondérations interfèrent avec la valeur des notes. Enfin, un comportement « intelligent » devrait avoir tendance à surnoter les options qui lui semblent intéressantes et sous-noter les autres afin d'augmenter les chances de voir sa préférence retenue. Or ce n'est ni souhaitable, ni simple à réaliser. Les difficultés associées aux votes sont nombreuses, nous y reviendrons dans le chapitre III.

4.4 Objectifs

Au vu des avantages et inconvénients du vote, nous nous proposons d'utiliser cette méthode de coordination d'actions en améliorant les points suivants.

Le premier axe concerne l'espace d'actions. Nous avons vu que restreindre les possibilités alors qu'il en existe une infinité n'est pas souhaitable. C'est pourquoi nous utiliserons un domaine continu pour l'espace d'actions.

Le second axe d'amélioration concerne la question de la granularité des comportements. Elle doit être considérée en même temps que les pondérations. Les agents doivent être de granularité plus fine.

Enfin, afin de limiter l'influence des pondérations et des notes et ainsi assurer la simplicité et la reproductibilité des choix, nous proposons de tester une autre procédure de vote.

5. Conclusion

La sélection d'actions est un problème complexe et récurrent. Les environnements continus et imprévisibles sont difficilement traités par les modèles décisionnels actuels. Les choix effectués par ces modèles sont rarement optimaux et leurs qualités sont d'ailleurs rarement quantifiables. Ces limitations ont conduit les chercheurs [Maes, 1989] [Tyrell, 1993] à proposer neuf critères permettant de comparer les modèles décisionnels.

Les architectures d'agents constituent les plans qui déterminent la manière dont sont construits les modèles décisionnels. L'approche symbolique classique souffre de plusieurs limitations particulièrement pénalisantes dans le cadre des environnements continus et dynamiques. Cette approche n'est donc pas totalement adaptée à l'animation comportementale. Les architectures comportementales sont relativement récentes et constituent une alternative intéressante à l'approche classique. Leur construction distribuée permet d'associer différents comportements. L'association de comportements réactifs et délibératifs, dirigés par des buts ou par l'environnement doit permettre de satisfaire la majorité des contraintes inhérentes aux modèles décisionnels. Le mécanisme de coordination d'actions est le point central des architectures distribuées. Les deux principales méthodes de coordination sont l'arbitrage et la fusion d'actions qui ont été abordées dans les partie 2 et 3.

La quatrième partie de ce chapitre synthétise les idées présentées lors de l'étude bibliographique. Le vote est une méthode très intéressante, en particulier parce qu'elle garantit que la majorité des comportements est satisfaite par l'action sélectionnée. Il apparaît néanmoins que ces architectures peuvent être améliorées afin de prendre en compte des actions continues, de diminuer la granularité des comportements et de réduire l'influence des pondérations et des notes. Le chapitre suivant est consacré aux propositions que nous faisons pour réaliser ces améliorations.

Chapitre III : Proposition d'un modèle décisionnel fondé sur le vote

Le chapitre deux nous a permis de détailler les avantages et les inconvénients des différents modèles décisionnels. Les architectures orientées comportement et plus particulièrement la politique du vote de préférences distribuées a particulièrement retenu notre attention car elle possède les avantages suivants : le choix est un bon compromis (Critère n°6), elle permet aux comportements d'exprimer leurs préférences et leurs contraintes (Critère n°2, situé) et s'adapte à différentes applications (Critère n°7). En revanche, nous avons mis en avant quelques défauts qu'il convient de traiter et que nous regroupons en trois principaux thèmes :

- Les méthodes de vote actuellement proposées dans le domaine ne sont pas satisfaisantes, en particulier parce qu'elles accordent trop d'importance aux notes attribuées ainsi qu'aux coefficients de pondérations des votes des comportements.
- La granularité des comportements nous apparaît souvent trop élevée par rapport à la volonté de distribution relative à l'approche orientée comportement.
- L'utilisation d'un espace d'action discret pose un problème de sélection des valeurs discrètes qui seront retenues et qui ne sont pas toujours adaptées.

Dans la première partie de ce chapitre nous présenterons les solutions envisagées pour pallier les défauts précédemment cités. Les solutions retenues nous amèneront à proposer notre propre architecture orientée comportement lors de la seconde partie de ce chapitre. Enfin, la troisième et dernière partie comporte une description du fonctionnement de l'architecture et notamment la spécification de l'algorithme de sélection d'actions distribuée.

1. Vers une amélioration du vote de préférences distribuées

Dans les paragraphes suivants, nous nous attacherons aux trois objectifs que nous avons mis en avant (la méthode de vote, la granularité et l'espace d'action continu). Nous commencerons par présenter la théorie du vote ainsi que les principales méthodes existantes. Ensuite, nous détaillerons comment nous souhaitons diminuer la granularité des comportements, ainsi que ses effets sur le reste du modèle décisionnel. Enfin, nous aborderons les avantages et difficultés relatives à la sélection d'une action dans un domaine continu.

1.1 Étude du mécanisme de sélection par vote

Le mécanisme de vote mérite une attention particulière car il est au centre du modèle décisionnel de l'agent : la qualité des choix effectués par un agent dépendant directement de la qualité de ce mécanisme. Le processus de vote doit prendre en compte les préférences individuelles de chaque comportement et les agréger en une préférence collective. Or, l'agrégation de préférences par le vote constitue un problème complexe, très étudié et dont l'enjeu dépasse de très loin les architectures orientées comportement.

Comme le rappelle [Hudry, 2003] ou [Boursin, 1999], la théorie du vote met en avant bon nombre de paradoxes allant à l'encontre de propriétés que l'on tient généralement pour acquises. Borda [Borda, 1781][14] ou Condorcet [Condorcet, 1785][14] avaient déjà au 18ème siècle mis en avant quelques uns de ces paradoxes. « L'effet Condorcet » montre l'irrationalité éventuelle d'un choix collectif : il est en effet possible que le choix A soit collectivement préféré à B, que B soit préféré à C et que C soit lui même préféré à A. Ces situations paradoxales ne se produisent pas systématiquement mais [Arrow, 1951][14] a prouvé qu'aucun système électoral ne peut respecter simultanément les trois principes suivants :

- Principe d'**unanimité** (ou principe de Pareto) : si un candidat est strictement préféré par tous les votants à un autre candidat, alors il doit aussi être strictement préféré à celui-ci dans la préférence collective.
- Principe de **non-dictature** : un individu ne peut dicter ses choix à la collectivité.
- Principe d'**indépendance** vis-à-vis des états non pertinents : le classement social entre deux possibilités ne dépend que des classements individuels entre ces 2 possibilités

Le théorème d'Arrow montre qu'il n'est pas possible de créer une fonction de choix social qui respecte ces trois principes simples. Toute procédure de vote existante s'autorise donc à violer quelques contraintes. La difficulté du choix social (fondé sur ce théorème) réside dans le fait qu'il montre uniquement ce qui est impossible et non ce qui est possible ou souhaitable.

Dans le cadre de notre problème de sélection d'actions nous souhaitons respecter en priorité un autre principe, celui d'universalité. Ce principe énonce que toutes les préférences doivent être prises en compte, or ce n'est pas le cas avec un espace d'actions discret.

1.1.1. Principales méthodes de vote

Nous allons décrire successivement quelques unes des méthodes de vote existantes. Nous ne prétendons pas à l'exhaustivité : il s'agit de méthodes les plus décrites ([Hudry, 2003]

14 Cité dans [Hudry, 2003]

[Boursin, 1999]). Notons que, pour un même exemple, chacune de ces méthodes peut donner un résultat différent.

Vote majoritaire

Nous pratiquons couramment le vote au scrutin majoritaire à un ou deux tours. Le scrutin majoritaire à un tour consiste simplement à choisir l'option (ou le candidat) qui recueille le plus de voix. Le scrutin majoritaire à deux tours permet de dégager un consensus. Mais l'élimination de candidats entre le premier et le second tour peut aussi conduire à l'élimination d'un candidat qui aurait pu recueillir un meilleur consensus.

Méthodes de vote par pondération

Les méthodes par pondération permettent à l'électeur de s'exprimer sur l'ensemble des options en leur attribuant une note. Parmi les méthodes de vote par pondération on retrouve la méthode par estimation moyenne qui est utilisée dans les architectures orientées comportement. La méthode par estimation moyenne consiste pour chaque votant à attribuer une note à chaque option. Cette note appartenant à un intervalle déterminé ([0 ; 1] pour le vote de préférences distribuées ou par exemple [0 ; 20] pour le classement des étudiants par l'ensemble de leurs enseignants). De même, il existe des méthodes de pondération ternaire qui restreignent les notations à 3 possibilités : pour, contre ou sans opinion. Le principal défaut de cette méthode est sa facilité à être manipulée[15]. En effet, les électeurs peuvent facilement (et en ont d'ailleurs intérêt) surnoter leur favori et sous-noter les autres.

Méthodes de vote par classement

Les méthodes par classement permettent également de s'exprimer sur l'ensemble des options candidates non plus, comme pour le vote par pondération en les notant, mais en les classant selon l'ordre de ses préférences[16]. Les méthodes par classement les plus connues sont la méthode de Condorcet, le vote alternatif et la méthode de Borda.

Pour Condorcet [Condorcet, 1785][17], le décompte des votes s'établit comme suit. Pour chaque paire de candidats, on détermine lequel est préféré en utilisant leurs classements respectifs sur chaque bulletin de vote. Si un candidat est préféré à tous ses adversaires, il remporte le vote. Il est possible qu'aucun des candidats ne satisfasse ce critère, mais s'il existe son choix satisfait l'ensemble des électeurs. La méthode de Condorcet n'aboutit pas nécessairement, dans ce cas il faut recourir à une autre méthode. Les probabilités qu'il n'y ait pas de vainqueur de Condorcet ou de la présence de « l'effet Condorcet » ne sont pas négligeables, en particulier lorsque le nombre d'alternatives est supérieur au nombre de votants (cf. [Hudry, 2003]). Or cela est courant dans un problème de sélection d'actions.

Le vote alternatif est un processus itératif. A chaque pas, on compte les voix des candidats premiers de liste. Si un candidat obtient la majorité absolue des voix, il est élu et le processus est terminé. Sinon, le candidat qui a recueilli le moins de voix est éliminé. Le candidat éliminé est rayé des bulletins, les rangs des candidats placés après lui sont donc modifiés. Le processus est réitéré jusqu'à ce qu'une majorité absolue se dessine, la méthode aboutit nécessairement.

15 Une méthode de vote est manipulable si les électeurs ont intérêt à ne pas exprimer leurs véritables préférences. Toute méthode de vote non dictatoriale étant manipulable selon [Hudry, 2003].

16 Dans la bibliographie, on parle d'ordre total ou de préordre total si le classement peut contenir des ex æquo.

17 Cité dans [Hudry, 2003]

La méthode de Borda [Borda, 1781] est fondée sur l'attribution de points aux candidats. Il existe plusieurs variantes de cette méthode : échelle de points croissante (choix du score minimal) ou décroissante (score maximal). Il est également possible d'utiliser une échelle n'ayant pas une progression arithmétique. Ce système de bonus (cf. partie gauche du tableau III.1) avantage les candidats classés favoris par plusieurs électeurs et désavantage les candidats qui obtiennent des classements moyens mais constants. Les points recueillis par les options sont additionnés et l'option dont le score est le moins élevé est choisie. Cette méthode est plus rapide que celle de Condorcet mais elle n'en vérifie pas le critère qui énonce que le choix est préféré à toutes les autres options. Il est possible que des options obtiennent le même nombre de points, dans ce cas, il faut définir un critère pour les départager.

Tableau III.1 : *Variantes de la méthode de Borda, exemples de points attribués en fonction des classements*[18].

Place	***Méthode des points minimaux***	***Méthode des points avec bonus***
1	0	0
2	1	3
3	2	5.7
4	3	8
5	4	10
6	5	11.7
7	6	13
+7	Ajouter un point	Ajouter un point

1.1.2. Discussion sur les méthodes de votes

Le vote majoritaire est l'une des méthodes les plus simples mais elle ne favorise pas le consensus ; le vote par pondération a déjà été utilisé dans les architectures orientées comportement [Sukthankar, 1997] [Rosenblatt, 1996] [Hostetler et al., 2002] avec les inconvénients que nous avons déjà présentés (en particulier les problèmes de mise au point et son interférence avec les coefficients associés aux comportements). C'est pourquoi nous choisissons d'adopter une méthode par classement.

Lors d'un suffrage par classement, les comportements transmettent simplement une liste ordonnée de leurs préférences au mécanisme de vote. Si les comportements sont développés de cette manière, il est donc possible de les utiliser avec toutes les méthodes de vote par classement (Condorcet, Borda, vote alternatif...).

Les méthodes par classement utilisent le plus souvent un ordre de classement strict, c'est-à-dire qu'une option est obligatoirement préférée à une autre ; il n'est pas possible de classer des options *ex aequo*. Or, nous estimons que cette éventualité est courante. La méthode de Borda offre cette possibilité. Par exemple, dans certaines variantes de la méthode, un électeur peut choisir de ne pas classer certaines options. Dans ce cas, les options non classées reçoivent le nombre maximal de points si on sélectionne le score le plus bas. (ou le nombre minimal si on sélectionne le score le plus élevé).

18 Le système avec bonus permet de favoriser les meilleures places. (Par exemple 2 places de second apportent plus de points qu'une place de 1er et un place de troisième) (Cet exemple est tiré des règles de course de International Sailing Fédération (ISAF), olympiade 2005-2008, c*f.* http://www.isaf.com)

La théorie met en avant les défauts de chaque méthode de votes mais ne peut indiquer ce qui convient pour notre application. Nous utilisons une variante de la méthode de Borda car bien que simple, elle aboutit nécessairement et permet d'utiliser des pré-ordres totaux. Conformément à cette méthode, des points sont accordés aux options en fonction de leur position dans le classement : 0 point si l'option est classée en tête, 1 point lorsqu'elle est seconde, 2 points pour une 3ème place, *etc.*

Le second axe d'amélioration de l'architecture de vote de préférences que nous souhaitons traiter concerne la granularité.

1.2 Vers une réduction de la granularité des comportements

La granularité des comportements a une influence sur les choix pris par l'architecture. Une granularité faible permet de mieux prendre en considération l'environnement (conformément à l'approche orientée comportement).

Nous souhaitons donc mettre en oeuvre des comportements de granularité faible quelle que soit l'application. Pour cela, nous choisissons de ne plus associer les comportements à des aspects généraux du problème mais à des aspects très spécifiques. Cela permet une distribution plus fine des tâches. La baisse de granularité des comportements s'accompagne nécessairement d'un accroissement de leur nombre. Pour cela, nous n'hésitons pas à dupliquer les comportements comme l'illustre la figure III.1 ci-dessous. Sur cette figure les comportements C3-i représentent plusieurs instances du même comportement C3 (de même pour C4). Chaque instance de ces comportements est associée à une instance d'une classe d'objets que l'agent doit prendre en compte pour effectuer son choix. Naturellement, il est également possible que certains comportements (C1 et C2 sur la figure) ne soient instanciés qu'une seule fois car ils correspondent à un but ou un événement unique.

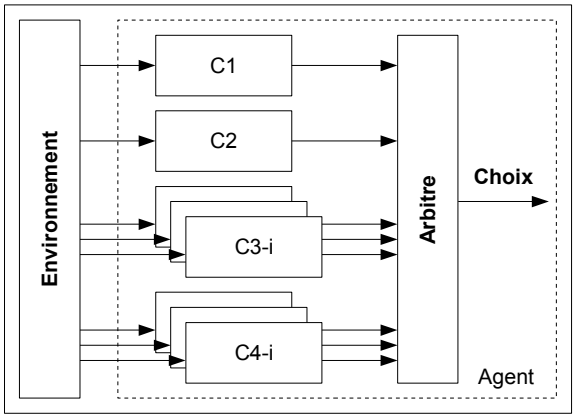

Figure III.1 : Granularité « plus fine » et duplication des comportements.

Pour être plus concret, reprenons l'exemple du modèle décisionnel de piéton virtuel présenté sur la figure II.19 (chapitre II, page 56). Sur cet exemple, le modèle décisionnel dispose d'un unique comportement « éviter les piétons ». Afin de réduire la granularité des comportements, nous proposons de le remplacer par un comportement « éviter le piéton P_i », instancié autant de fois qu'il y a de piétons à prendre en compte dans l'environnement proche de l'agent.

Le fait de dupliquer les comportements peut avoir un impact positif sur l'utilisation des pondérations. En effet, leur utilisation n'est plus justifiée car les comportements sont tous au même niveau d'abstraction. La réduction de granularité des comportements nous encourage

donc à ne pas utiliser de coefficients de pondération. Dans l'exemple précédent, la présence de nombreux piétons n'est pas représentée par une forte pondération, mais par une augmentation du nombre de comportements d'évitement. Il y a néanmoins une contrepartie à cette proposition. Le nombre des comportements devient variable et la structure décisionnelle de l'agent doit être adaptée au contexte. Il faudra donc instancier les comportements en fonction des perceptions de l'agent. Cette fonction qui associera un comportement à chaque type de perceptions sera nécessairement dépendante de l'application considérée. Nous revenons sur ce point dans le paragraphe 3 (Proposition d'un mécanisme de sélection distribuée).

Les architectures fondées sur le vote réalisent la sélection d'actions dans un espace discret. Cette solution nous semblant trop restrictive, nous considérons ce problème dans le paragraphe suivant.

1.3 Vers une sélection d'actions dans un domaine continu

Nous souhaitons mettre en oeuvre une procédure de sélection d'actions distribuée fondée sur le vote. Les architectures présentées sélectionnent les actions dans un ensemble discret et prédéfini de neuf actions [Suhkthankar, 1997] [Hostelter et al., 2002]. Or, limiter les possibilités d'un agent évoluant dans un environnement continu à neuf actions, nous paraît beaucoup trop restrictif et arbitraire.

Nous pourrions envisager d'augmenter le nombre des possibilités considérées. Cette solution n'est pas satisfaisante et ne change rien sur le fond du problème. En effet, on présuppose toujours que les alternatives seront nécessairement adaptées aux situations auxquelles seront confrontées les agents. Le modèle décisionnel devra donc évaluer de nombreuses alternatives dont rien ne garantit qu'elles soient pertinentes.

Nous avons donc décidé de supprimer l'espace d'action discret et prédéfini. De même que les membres d'une assemblée proposent les alternatives qui seront soumises au vote, les comportements proposeront eux-mêmes les alternatives soumises à la sélection. A chaque cycle, les comportements pourront proposer autant d'alternatives que nécessaire.

Le choix dans un espace continu est, certes plus complexe, car le nombre de possibilités est infini, mais apporte de nombreux avantages. En particulier, il n'est plus nécessaire pour le concepteur de définir un ensemble de possibilités. De plus, laisser les comportements soumettre les options candidates au vote permet de respecter le principe d'universalité car toutes leurs préférences seront prises en compte (dans les architectures originales, le choix est tronqué avant l'élection). Ainsi les alternatives étant plus pertinentes, le choix de l'agent peut être mieux adapté au contexte dans lequel il évolue.

En revanche, la complexité du mécanisme de sélection d'actions s'en trouve augmenté. Il faudra donc veiller à ce que le nombre d'options proposées ne soit pas inutilement élevé. De même, il conviendra de surveiller que le temps d'exécution du modèle reste compatible avec l'application considérée.

Pour réaliser les changements proposés, les notions d'options et veto utilisées dans les architectures doivent être redéfinies. De même les comportements seront adaptés afin qu'ils proposent plusieurs alternatives. La procédure de vote n'est pas affectée par ces changements : le vote intervenant après que les alternatives aient été soumises, il ne dépend donc pas de l'origine de ces alternatives (proposées par les comportements ou fixées par le concepteur). Les détails du modèle concernant les options, veto et comportements sont développés dans la partie suivante du chapitre.

2. Proposition d'une architecture orientée comportement à coordination par vote

Les objectifs et les solutions retenues nous amènent à redéfinir plusieurs éléments des architectures utilisant le vote présentées au chapitre II.

Dans le premier paragraphe de cette partie nous décrirons les comportements utilisés par notre proposition d'architecture. Nous décrirons en particulier les mécanismes suivant lesquels le choix (perception, décision, action) est réparti entre les comportements. Puis nous définirons les données qu'ils manipulent et particulièrement comment ils expriment leurs réponses, contraintes ou préférences.

Enfin, nous aborderons la manière dont les comportements s'articulent entre eux au niveau de l'architecture d'un agent.

2.1 Comportements.

Le comportement constitue l'élément de base d'une architecture. Bien que les comportements soient adaptés en fonction de l'application, il convient de les définir selon un modèle qui permet de les assembler d'une manière cohérente. Nous adaptons le modèle classique des comportements (présenté dans le chapitre II) pour répondre aux objectifs que nous nous sommes fixés.

2.1.1. Modèle de comportement

Comme le montre la figure III.2, un comportement est composé de trois fonctions principales : percevoir, proposer, évaluer. Un comportement dispose également de deux entrées (les stimuli de l'environnement et les réponses des autres comportements) et de trois sorties (correspondant à l'expression des réponses, contraintes et préférences).

Comme pour toutes les architectures orientées comportement, la perception est distribuée entre les comportements. La perception d'un comportement est alimentée par des stimuli spécifiques provenant de l'environnement.

Pour les autres architectures à base de vote, les options considérées par les comportements pré-existent dans l'espace d'actions, indépendamment des comportements. Cela n'est pas souhaité dans notre architecture. Les options sont donc proposées par les comportements eux mêmes. Le nombre d'options proposées par comportement n'est pas limité mais il faut garder à l'esprit qu'il ne doit pas être trop élevé pour éviter un temps de raisonnement trop long.

Le problème est similaire pour les veto. Pour [Sukthankar, 1997], un veto est associé à une unique option dans l'espace d'actions discret. Il conviendra donc de redéfinir la notion de veto. Un veto s'appliquera à un ensemble d'options et non plus à une valeur unique. De même que pour les options, les comportements peuvent proposer en sortie autant de veto qu'ils le souhaitent afin d'exprimer leurs contraintes.

La seconde entrée d'un comportement (figure III.2) provient du mécanisme de coordination et correspond aux propositions faites par les autres comportements. Ces propositions « alimentent » la fonction d'évaluation car, comme nous l'avons rappelé, l'ensemble des options envisagées n'est plus défini dans l'espace d'actions. La manière dont les comportements expriment leurs préférences est relative aux mécanismes de vote utilisés. Nous

proposons que les comportements transmettent leurs préférences sous forme d'un classement des options. Cela permet d'utiliser différentes méthodes de vote et, en particulier, celle de Borda [Borda, 1781][19] que nous avons retenue. En effet, un classement n'équivaut pas à une notation mais peut être utilisé pour (par exemple) réaliser un vote majoritaire. Les classements sont d'ailleurs utilisés dans la bibliographie pour présenter les différences entre plusieurs méthodes de vote.

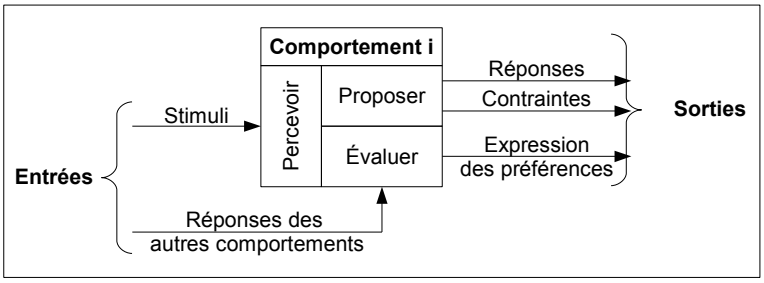

Figure III.2 : Comportement utilisé par l'architecture.

Le fait que les comportements proposent des options puis les évaluent constitue une différence par rapport aux autres architectures. Pour celles à base de vote, les comportements ne proposent rien mais évaluent les possibilités. Pour celles qui utilisent la combinaison d'actions, chaque comportement fait une proposition mais l'évaluation n'est pas distribuée. Notre proposition présente donc l'avantage de mieux distribuer la sélection d'actions en faisant intervenir les comportements dans plusieurs de ses étapes.

Notre approche des notions d'option et de veto est décrite dans les paragraphes suivants.

2.1.2. Réponses des comportement : les options.

Les options sont représentées sous forme d'un vecteur similaire à la fusion d'actions (cf. [Arnaud, 2000]). Les composantes des options correspondent[20] avec celles du vecteur de commande v_t (chapitre II). A la différence de la combinaison d'actions, les valeurs des composantes ne sont pas nécessairement réelles. De plus, elles ne correspondent pas nécessairement à des composantes de forces (répulsion, attraction) essentiellement utilisables pour la gestion de déplacements (de robots mobiles [Arkin, 1989], d'animats [Flacher et al., 2003] ou d'avions [Zeghal, 1993]).

Exemple

Considérons l'exemple des robots fourrageurs bien connu en systèmes multi-agents, cf. par exemple [Ferber, 1995] ou [Grislin et al., 1996]. Un robot foreur doit gérer, par exemple, outre ses déplacements (direction et vitesse), son burineur {Marche, arrêt}), la vitesse de rotation du foret {Arrêt, lent, rapide} et ses messages { Je fore, je ne peux pas accepter d'autres requêtes. », « Je ne fore pas, Je peux accepter d'autres requêtes. », « J'ai besoin d'un transporteur. »}

v_t = (direction, vitesse, burineur, rotation_foret, message)^T

19 Cité dans [Hudry, 2003].

20 Ou au moins avec une partie de ce vecteur. Le modèle décisionnel pouvant être réparti entre plusieurs architectures comportementales distinctes comme dans [Rosenblatt et al., 2002].

Joker

Nous avons souhaité qu'un comportement puisse éventuellement ne pas définir entièrement sa proposition. En conséquent, nous proposons d'ajouter au domaine de définition d'une composante, le symbole joker « * ». Par exemple, dans le cadre de déplacements autonomes, il est possible qu'un comportement propose une direction pour contourner un obstacle. En revanche, la vitesse peut ne pas être importante de son point de vue. Celle-ci ne sera donc pas développée dans sa proposition et un joker viendra prendre la place de la composante de vitesse dans les options proposées.

Il faut noter qu'aucune architecture ne propose la notion de « joker ». D'une part, parce qu'il ne serait pas possible de fusionner des actions non spécifiées. D'autre part, pour le vote de préférences distribuées, l'espace d'action étant limité (cf. tableau II.4 au chapitre II), il suffit de noter une même ligne ou une même colonne de manière identique. Naturellement, le fait d'utiliser des options qui ne sont pas entièrement définies influe sur le mécanisme de sélection d'actions proposé. Nous aborderons cette question dans le paragraphe 3.4 de ce chapitre.

Nous proposons une formulation des options pour prendre en compte ce que nous venons de présenter.

Formulation

Conformément à la définition d'une action élémentaire de l'agent (le vecteur de commande v_t), une option o_i est caractérisée par un vecteur dont chaque composante $o_{i,j}$ correspond à une variable de commande $v_{t,j}$ et appartient à un domaine de définition D_j inclus dans l'ensemble des réels ou tout autre domaine continu, défini par intervalles ou fini. Une option o_i est donc un vecteur défini comme suit :

$o_i=(o_{i,1};o_{i,2};...;o_{i,n})$ avec $o_{i,j}\in D_j\cup\{*\}$ (formule III.1)

Les veto sont redéfinis de la même manière.

2.1.3. Contraintes des comportements : les veto

Introduit par [Sukthankar, 1997] dans le cadre des architectures orientées comportement, le veto permet à un comportement de s'opposer à la sélection d'une action qui lui semblerait trop néfaste. Pour [Sukthankar, 1997], le veto est associé à une option particulière. Afin de maintenir une certaine unité, nous préférons qu'un même veto soit associé à un ensemble de valeurs. Nous optons pour un veto sous forme de vecteur d'intervalles.

Joker

Comme pour les options, une composante d'un veto peut ne pas avoir d'importance. Dans ce cas, elle est également notée « * ». A la différence des options, il est possible d'assigner une valeur à la composante notée « * ». Cette valeur sera le domaine de définition D_j de la variable de commande associée. De cette manière, ce sont les autres composantes du veto qui seront déterminantes pour la sélection d'actions (un exemple sera détaillé par la suite).

Cette notion de joker sur un veto n'existe pas dans SAPIENT [Sukthankar, 1997], le nombre d'actions est très limité (neuf cas), il est possible de s'opposer à toutes les options d'une même ligne ou d'une même colonne de l'espace d'actions (cf. chapitre II) en déposant seulement trois veto.

Cela nous amène à formuler les veto comme suit.

Formulation

La définition des veto est similaire de celle des options, elle est exprimée par la formule suivante. La valeur d'un veto v_k est un vecteur d'intervalles, chaque intervalle appartenant au domaine de définition de la variable de commande correspondante.

$$v_k = (]v_{k,1}; v_{k,1}'[;]v_{k,2}; v_{k,2}'[; \ldots;]v_{k,n}; v_{k,n}'[) \text{ avec } v_{k,j} \in D_j \quad \text{(formule III.2)}$$

La composante d'un veto peut ne pas avoir d'importance. Le comportement qui dépose le vote note sa composante « * » mais sa valeur est traduite ainsi : $]v_{k,j}; v_{k,j}'[= D_j$

Exemple

Si le modèle décisionnel est utilisé pour gérer les déplacements (direction et vitesse) d'un agent mobile, les domaines de définition des variables de commande sont les suivants : $D_1 =]-180; 180]$ et $D_2 = [0; 20]$.

La notion de veto peut être utilisée pour :

- interdire de se diriger dans un secteur angulaire en déposant le veto :
  $v_1 = ([10; 30]; *)$
- limiter la vitesse en déposant le veto :
  $v_2 = (*; [10; 20])$
- limiter la vitesse dans une direction donnée :
  $v_3 = ([10; 20]; [10; 20])$

A titre d'exemple, et afin d'illustrer la présentation des comportements et des données qu'ils manipulent, nous proposons de décrire un comportement particulier : l'inertie (ou *hystérésis*) [Sukthankar, 1997].

2.1.4. Exemple d'un comportement particulier : l'inertie

Nous pouvons utiliser notre modèle pour réaliser un comportement d'« inertie » tel que présenté par [Sukthankar, 1997] ou [Hostetler et al., 2002]. Cet exemple semble pertinent car il est suffisamment simple et générique pour être décrit ici. De plus, il pourra être réutilisé pour diverses applications. Tout comme le fait Sukthankar, de nombreuses possibilités peuvent être imaginées pour implémenter ce comportement.

Le module « percevoir » du comportement *inertie* ne pose pas de problème particulier puisque la seule chose que ce comportement prend en compte est le choix réalisé au pas précédent de la simulation.

De même, il suffit au module « proposer » de reprendre l'option qui avait été retenue précédemment. Le droit de veto de ce comportement pourrait être utilisé pour imposer une limite dans la variation des choix (ce que font d'ailleurs certaines architectures qui filtrent les variations trop importantes cf. par exemple [Rosenblatt, 1996]).

Comme les autres comportements, *inertie* doit exprimer ses préférences en classant les options proposées par les autres comportements. Il préférera les options (valides) qui sont proches de l'ancien choix. Il va donc classer en tête sa propre proposition, puis celle qui lui est la plus semblable, etc. Nous revenons sur ce critère de préférence dans le paragraphe 3.6 (Etape 5, évaluer les options). La figure III.3 résume la présentation du comportement *inertie*.

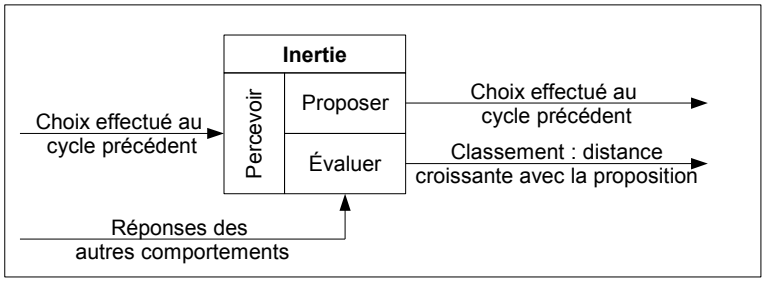

Figure III.3 : Comportement *inertie*.

Nous avons présenté les comportements et les données qu'ils génèrent. Nous proposons, lors du paragraphe suivant de présenter de quelle manière ceux-ci s'articulent dans le modèle décisionnel d'un agent.

2.2 Architecture

La manière dont les comportements sont associés au sein de l'architecture est présentée à la figure III.4. Cette architecture est adaptée de la bibliographie [Rosenblatt, 1996] [Sukthankar, 1997] et [Hostetler et al., 2002].

Elle met en oeuvre un nombre de comportements variant au cours du temps, ce qui est la conséquence de la baisse de granularité des comportements. En effet, ceux-ci étant associés à des objets ou événements précis, sont instanciés ou retirés du modèle décisionnel en fonction de l'évolution de l'environnement pris en compte par l'agent. (Une première étape de perception « grossière » est donc réalisée pour instancier les comportements mais n'apparaît pas sur la figure III.4.)

Chaque comportement propose des réponses (options) et exprime ses contraintes sous forme de veto. L'une des différences par rapport à la bibliographie est l'existence de bouclages : l'ensemble des options proposées par les comportements est transmis aux comportements pour être évalué.

La méthode de vote n'est pas imposée, excepté qu'elle doit permettre aux comportements d'exprimer leurs préférences sous forme de classement.

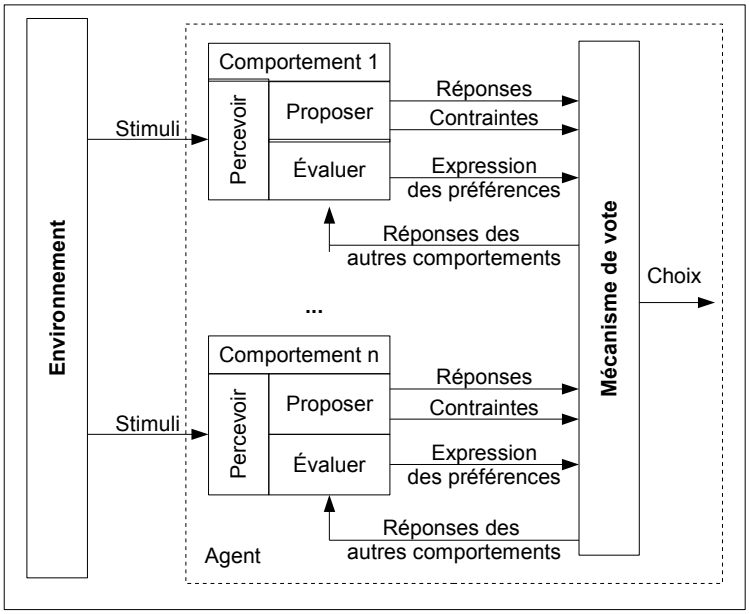

Figure III.4 : Adaptation des architectures orientées comportement fondées sur le vote.

L'architecture présentée impose un minimum de synchronisation entre les comportements. Par exemple, un comportement ne peut pas évaluer les options avant que celles-ci aient toutes été déposées par les autres comportements. La synchronisation est nécessaire pour le vote. Cependant, entre ces étapes de vote, les comportements peuvent toujours s'exécuter selon des cycles de vie indépendants.

Cela nous amène à définir l'algorithme selon lequel fonctionne l'architecture. Ce mécanisme général permettant à une telle architecture de sélectionner des actions est décrit dans le paragraphe suivant.

3. Proposition d'un mécanisme de sélection d'actions distribuée

Les objectifs fixés nous ont conduit à modifier les architectures dont nous nous inspirons ([Rosenblatt, 1996] [Sukthankar, 1997] et [Hostetler, 2003]) et notamment les comportements et les données qu'ils manipulent. L'architecture proposée est composée d'un ensemble variable de comportements distincts, lesquels doivent sélectionner collectivement une action pertinente vis-à-vis de la situation présente. Le principe général du mécanisme de sélection d'actions proposé est d'abord décrit, avant de détailler chacune des étapes qui le constituent.

3.1 Principe général

Le mécanisme de sélection d'actions proposé comporte six étapes présentées en figure III.5. L'ensemble des comportements se trouve au centre de la figure. Ces comportements ne sont pas détaillés car, sauf exception (telle que l'inertie par exemple), ils sont développés spécialement pour une application particulière. La signification de chaque composante d'actions (ou de veto) est partagée par l'ensemble des comportements et également par les effecteurs de l'agent.

L'étape 1 est spécifique à l'application car elle traite un ensemble de comportements définis par le concepteur du système, en fonction des actions à générer et de l'environnement. Les étapes 2 et 5 sont propres aux comportements, qui génèrent et évaluent les propositions en fonction de leurs objectifs individuels. Les étapes 3, 4 et 6 sont génériques car indépendantes de l'application ; elles consistent en des traitements numériques appliqués quelle que soit la signification des données manipulées.

Le cycle de fonctionnement de l'agent commence par une rapide perception de l'environnement. Ce traitement préalable permet d'établir « l'ensemble des objets intéressants » (en haut de la figure III.5). L'ensemble des perceptions est une liste d'objets de l'environnement proche devant être pris en compte.

Des comportements sont instanciés lors de l'étape 1 à partir de ces perceptions et des buts de l'agent. Etant donné que l'environnement et les comportements dépendent de l'application, cette étape dépend également de l'application pour laquelle le modèle est utilisé.

Lorsque les comportements ont été instanciés, il est possible de passer à l'étape 2. Celle-ci consiste simplement en l'exécution des fonctions « percevoir » et « proposer » de chacun des comportements instanciés. Cette étape est malgré cela générique puisqu'elle consiste uniquement à utiliser les comportements développés pour l'application. A la fin de l'étape 2, le mécanisme dispose de toutes les réponses et de toutes les contraintes des comportements.

Les propositions faites par les comportements peuvent contenir des options non spécifiées. C'est-à-dire que les options peuvent comporter un ou plusieurs jokers. Comme il n'est pas possible d'évaluer de telles options, celles-ci sont donc spécifiées à l'étape 3. L'algorithme permettant de compléter les options manipule les données numériques indépendamment de leur signification. Il ne dépend donc pas de l'application.

Il est possible que des veto aient été déposés. Le mécanismes de sélection d'actions doit alors éliminer les options qui ne respectent pas ces veto. Cette opération est réalisée lors de l'étape 4 qui, comme le montre la figure, considère les options spécifiées ainsi que les veto pour déterminer l'ensemble des options valides.

L'ensemble des options valides ayant été formé, il va être retourné aux comportements pour être évalué. Lors de l'étape 5 le mécanisme de sélection utilise donc les modules « évaluer » de chaque comportement en leur fournissant l'ensemble des options valides. Les comportements retournent leurs préférences au mécanisme de vote sous la forme d'un classement des options (conformément au paragraphe 1.1. sur le vote).

A la fin de l'étape 5, le mécanisme de sélection d'actions dispose de toutes les préférences des comportements vis à vis des options valides. Il peut donc utiliser toutes ses données lors de l'étape 6 afin de déterminer l'option la plus intéressante. Cette option sera finalement réalisée par l'agent.

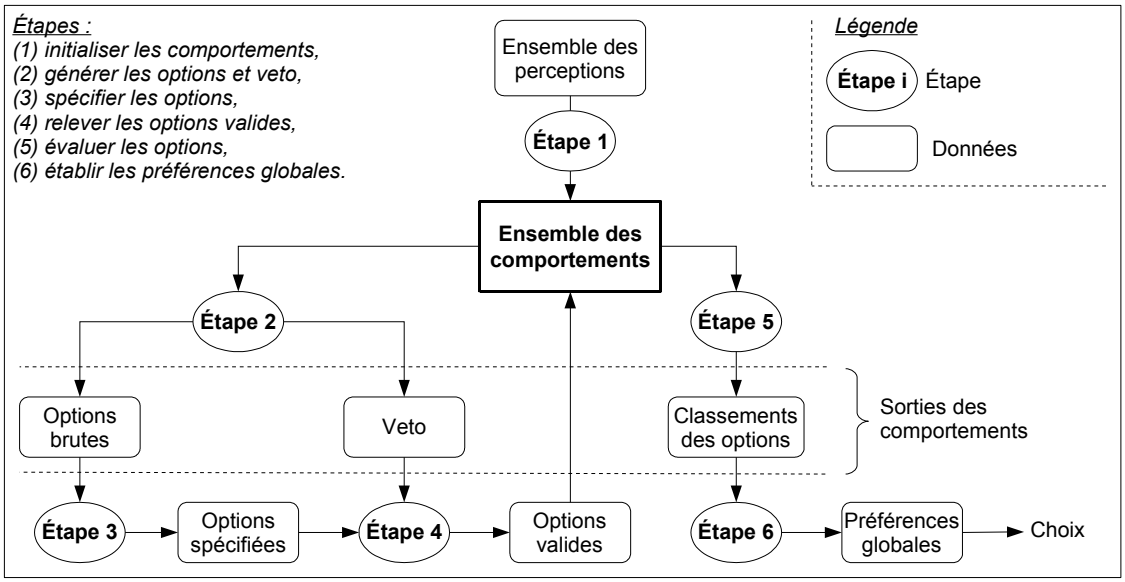

Figure III.5 : Proposition d'un mécanisme de sélection d'actions.

Mis à part les comportements qui, naturellement, sont développés pour une application particulière, seule l'étape 1 du mécanisme de sélection d'actions n'est pas générique. Les 6 étapes du mécanisme de sélection d'actions sont détaillées dans les paragraphes suivants :

(1) initialiser les comportements,
(2) recueillir les options et veto,
(3) spécifier les options,
(4) relever les options valides,
(5) évaluer les options,
(6) établir les préférences globales.

3.2 Etape 1 : percevoir et initialiser les comportements.

Nous avons exprimé notre souhait de voir la granularité des comportements réduite. Nous avons vu que cette réduction de granularité s'accompagne d'un accroissement du nombre de comportements. Certains comportements seront également dupliqués afin d'associer un comportement à un objet particulier. Comme l'environnement dans lequel évolue l'agent peut être dynamique, il sera donc nécessaire d'instancier dynamiquement les comportements. C'est l'objectif de cette première étape de l'algorithme.

En entrée de cette étape, l'algorithme dispose de la liste des objets d'intérêt. Il s'agit de faire en sorte qu'à chacun de ces objets corresponde une instance d'un des comportements réunis pour l'application courante.

Comme nous l'avons déjà signalé (cf. figure III.1), certains comportements sont toujours utiles au modèle décisionnel (par exemple un comportement dirigé par un but important). D'autres doivent être instanciés en fonction du contexte, en particulier les comportements dirigés par des événements. Cette mise à jour se déroule comme suit :

- Si le modèle décisionnel utilise des comportements ne possédant pas de mémoire, ceux-ci peuvent être ré-initialisés à chaque cycle. Il est donc envisageable de retirer tous les comportements de l'architecture puis d'instancier les comportements nécessaires uniquement en fonction des perceptions.
- Sinon il est nécessaire de préserver la continuité, entre les cycles, des comportements avec mémoire. La mise à jour sera donc sélective afin de retirer de l'architecture uniquement les comportements qui ne sont plus utiles et d'ajouter ceux qui seraient éventuellement devenus nécessaires.

A la fin de cette étape l'architecture comporte l'ensemble des comportements nécessaires à la prise en compte des principaux objets de l'environnement proche de l'agent.

Il est à noter que cette partie de l'algorithme ne peut pas être générique puisqu'elle dépend pleinement de l'application. Il s'agit en effet d'associer des comportements et des objets qui lui sont spécifiques. Si on souhaite pourtant rendre cette partie générique, il faut que les comportements utilisés déclarent leurs compétences afin que le modèle décisionnel puisse les recruter automatiquement. (Cela existe dans certains systèmes multi-agents où certains agents coordonnateurs recrutent d'autres agents.)

3.3 Etape 2 : générer les options et veto.

L'ensemble des comportements utilisés par le modèle décisionnel a été mis à jour lors de l'étape précédente. Il convient dans l'étape présente de recueillir les options et veto proposés par les comportements et correspondant à l'expression de leurs préférences et contraintes.

Les techniques mises en oeuvre pour obtenir ces propositions et contraintes peuvent être diverses, même au sein d'un même modèle décisionnel : des plus simples (stimulus réponse) aux plus complexes (système à base de connaissances). De même qu'un agent peut être lui même composé d'un système multi-agents[21], un comportement peut lui même être bâti sur une architecture comportementale.

Rappelons également que nous ne limitons pas le nombre d'options et de veto que propose un comportement. Cela constitue un avantage puisque chaque comportement peut exprimer les différentes possibilités qui sont intéressantes de son point de vue. Ce peut être aussi un inconvénient car la complexité du mécanisme de sélection d'actions est naturellement fonction du nombre d'options et de veto déposés.

Le mécanisme de sélection d'actions est présenté ici de façon séquentielle mais ce n'est pas obligatoire. En effet, les comportements peuvent être exécutés en parallèle, voire sans aucune forme de synchronisation (moyennant quelques aménagements comme dans [Rosenblatt, 1996]). Ce point n'étant pas primordial à ce stade nous n'avons pas étudié davantage la question.

3.4 Etape 3 : spécifier les options brutes

Des options et veto ont été déposés à l'étape précédente. Or, nous permettons aux comportements de ne pas spécifier entièrement les options. Par exemple, dans le cadre de déplacements autonomes, un comportement peut proposer une option dont il spécifie la direction mais pas la vitesse (car elle n'a pas d'importance pour lui). Cependant, il est impossible d'évaluer une option qui ne serait pas entièrement spécifiée, sauf pour quelques cas particuliers (cf. étape 5, évaluer les options). La présente étape va donc permettre de spécifier les éventuelles options incomplètes.

L'objectif est donc d'obtenir un ensemble d'options dont toutes les composantes sont différentes du joker « * ». C'est-à-dire que la condition suivante est respectée.

$$\forall o_i=(o_{i,1},\ldots,o_{i,n}),\forall j, o_{i,j}\neq * \quad \text{(formule III.3)}$$

Nous proposons un algorithme pour spécifier les options. Le principe de cet algorithme est d'utiliser les composantes d'options spécifiées pour remplacer les composantes non spécifiées. Puisque la valeur « joker » signifie une indifférence vis-à-vis de la valeur de la composante correspondante, il est intéressant de considérer les propositions faites par les autres comportements en vertu du critère n°6 de bon compromis (cf. chapitre II). Le principe est donc de créer autant de déclinaisons de l'option initiale (avec joker) que de valeurs distinctes proposées par les autres comportements en remplacement de cette valeur manquante.

Les détails de cet algorithme sont donnés dans le paragraphe suivant. L'algorithme peut être utilisé avec des options à n composantes. Voici deux exemples pour n = 2 et n = 3. Dans l'exemple 1 ci-dessous, o_{03} est une option dont la première composante est spécifiée (valeur 4) mais pas la seconde. Pour spécifier cette option nous utilisons les deux valeurs spécifiées pour

21 Selon le principe de récursion [Demazeau, 1995]

la seconde composante (valeurs 6 et 7 de o_{01} et o_{02}). Cela donne deux options spécifiées o_{13} et o_{14} composées de la première composante de o_{03} et de la seconde de o_{01} et o_{02}.

Exemple 1. Soit la liste d'options brutes :

$o_{01} = (2 ; 6)$
$o_{02} = (3 ; 7)$
$o_{03} = (4 ; *)$

Après spécification, la liste suivante est obtenue.

$o_{11} = (2 ; 6)$ Option o_{01} déjà spécifiée.
$o_{12} = (3 ; 7)$ Option o_{02} déjà spécifiée.
$o_{13} = (4 ; 6)$ « Croisement » entre o_{03} et o_{01}.
$o_{14} = (4 ; 7)$ « Croisement » entre o_{03} et o_{02}.

Exemple 2. Soit la liste d'options brutes :

$o_{01} = (2 ; 3 ; 4)$
$o_{02} = (5 ; 6 ; *)$
$o_{03} = (7 ; * ; *)$

Après spécification, la liste suivante est obtenue.

$o_{11} = (2 ; 3 ; 4)$ Option o_{01} déjà spécifiée.
$o_{12} = (5 ; 6 ; 4)$ « Croisement » entre o_{02} et o_{01}.
$o_{13} = (7 ; 3 ; 4)$ « Croisement » entre o_{03} et o_{01}.
$o_{14} = (7 ; 6 ; 4)$ « Croisement » entre o_{03}, o_{02} et o_{01}.

L'algorithme que nous proposons pour compléter les options brutes est le suivant. Cet algorithme prend en entrée une liste d'options (listeOptions) dont certaines ne sont pas entièrement spécifiées. Il retourne une liste d'options entièrement spécifiées (listeOptionsSpec).

Algorithme 3.III.1 : *Spécification des options brutes.*

```
ListeOptionSpec ← ∅
Pour chaque composante j du vecteur d'options faire
   // 2 listes distinctes sont créées :
   // celles des options spécifiées et celles qui ne le sont pas
   Pour chaque option oi de listeOptions faire
      si (oi,j = * ) alors
         ajouter oi à listeOptionsIndj
      sinon
         ajouter oi à listeOptionsSpecj
      fin si
   fin pour
   //On crée une liste d'options spécifiées (listeOptionj)
   //à partir de chaque option de la listeOptionsIndj
   pour chaque option oi de listeOptionIndj faire
      listeOptionj ← ø
      pour chaque option ok de listeOptionSpecj faire
         créer option onew
         onew ← oi
         onew,j ← ok,j
         ajouter onew à listeOptionsj
      fin pour
   fin pour
fin pour
//la liste finale est créée en intégrant les listes spécifiques aux composantes
Pour chaque composante j
   listeOptionsSpec ← listeOptionsSpec U {listeOptionSpecj U listeOptionsj}
fin pour
retourner listeOptionsSpec
```

La méthode employée pour compléter les options augmente fortement leur nombre. La présence d'options non spécifiées influe donc sur le nombre d'options qui seront évaluées par le mécanisme de sélection d'actions. Dans les deux exemples, les trois options en entrée donnent quatre options en sortie. Il est donc important de maîtriser la quantité d'options non spécifiées. Dans les expérimentations, il sera possible de dénombrer le nombre maximal, minimal et moyen d'options brutes (en entrée) et spécifiées (en sortie). Cela nous permettra d'avoir un indice sur les temps d'exécution.

L'algorithme retourne une liste d'options spécifiées vide s'il existe au moins une composante pour laquelle aucune valeur n'a été spécifiée. La sélection d'actions ne peut être réalisée dans ce cas. Notons que ce cas apparaît rarement si l'agent utilise un comportement d'inertie qui propose la solution retenue au cycle précédent.

À la fin de cette troisième étape, une liste d'options spécifiées a été créée. Il convient à présent de commencer la sélection.

3.5 Etape 4 : relever les options valides

Cette étape est la première phase de sélection des options qui ont été proposées puis éventuellement « spécifiées » à l'étape précédente. Si des veto ont été déposés, il convient, lors de cette étape, d'éliminer les solutions qui ne les respectent pas.

Il s'agit donc, pour chaque option, de vérifier si elle est valide ou non. Une option n'est pas valide s'il existe au moins un veto pour lequel toutes les composantes de l'option appartiennent aux intervalles correspondants de ce veto. Si l'on note X l'ensemble des options valides, le processus de sélection peut se noter de la manière suivante :

$\forall o_i(\exists v_k(\forall j ; o_{i,j} \in [v_{k,j} ; v_{k,j}']) \Rightarrow o_i \notin X)$ (formule III.4)
avec X l'ensemble des options valides.

Nous utiliserons plus couramment le corollaire de la formule précédente : si, pour chaque veto, au moins une composante d'une option n'appartient pas à l'intervalle correspondant dans le vecteur de ce veto, alors cette option est valide.

$\forall o_i(\forall v_k(\exists j ; o_{i,j} \notin [v_{k,j} ; v_{k,j}']) \Rightarrow o_i \in X)$ (formule III.5)
avec X, l'ensemble des options valides.

Exemple d'application de veto sur un ensemble d'options

Pour être plus concret, considérons à nouveau l'exemple du paragraphe 3.1.3 (les variables de commandes correspondent à une direction et une vitesse). Les domaines de définition des composantes sont les suivants :

- $D_1 =]-180 ; 180]$
- $D_2 = [0 ; 20]$

On considère les deux veto v_1 et v_2 dont les valeurs sont :

- $v_1 = ([10 ; 30] ; *)$
- $v_2 = (* ; [10 ; 20])$

Les veto pris en exemple sont composés de jokers qu'il est nécessaire de remplacer par une valeur. Le veto v_1 interdit à l'agent de se diriger dans un secteur angulaire ; ce qui signifie que quelle que soit la vitesse d'une option si sa direction appartient à l'intervalle [10 ; 30] alors elle doit être éliminée. Cela est possible en remplaçant le joker par le domaine de définition correspondant. En remplaçant les jokers par les intervalles de définition correspondant, v_1 et v_2 deviennent :

- $v_1 = ([10 ; 30] ; [0 ; 20])$
- $v_2 = (]-180 ; 180] ; [10 ; 20])$

Soient les options :

- $o_1 = (20 ; 5)$; o_1 est éliminée par v_1 car $20 \in [10,30]$ et $5 \in [0,20]$
- $o_2 = (50 ; 15)$; o_2 est éliminée par v_2 car $50 \in]-180,180]$ et $15 \in [10,20]$
- $o_3 = (60 ; 5)$; o_3 reste valide car $60 \notin [10,30]$ et $5 \notin [10,20]$

Par conséquent, l'ensemble des options valides est : $X = \{o_3\}$. Dans cet exemple, il existe une option valide. Il est préférable qu'il en existe plusieurs mais il est également possible qu'aucune option ne soit valide. Cela peut se produire :

- Soit parce qu'aucune option acceptable n'a été proposée.
- Soit parce que les veto bloquent tout l'espace d'actions (le problème est sur-contraint).

L'algorithme d'apprentissage de [Sukthankar, 1997] pénalise les agents dont les comportements déposent des veto contre toutes les options. En effet, conformément au critère d'anticipation (cf. chapitre I), un bon modèle décisionnel aurait dû prévenir cette situation. Cependant, même après l'apprentissage, cette situation peut se reproduire. Peut-on développer une méthode qui permette de prévenir cette situation ou faut-il envisager une procédure à appliquer, le cas échéant ? Nous n'avons pas résolu ce problème dans le cadre de notre proposition, nous l'envisagerons dans les perspectives (chapitre V).

Cette étape 4 a permis d'éliminer les options qui ne respectent pas les veto et qui, par conséquent, auraient pu être néfastes pour l'agent. Dans de la suite de l'algorithme, cet ensemble d'options est évalué par l'ensemble des comportements afin d'opérer une sélection.

3.6 Etape 5 : évaluer les options

Les options ne respectant pas les veto ont été éliminées lors de l'étape précédente ; les options valides doivent, à présent, être évaluées par chacun des comportements, pour finalement en choisir une à l'étape 6.

Afin d'améliorer les performances du mécanisme de vote, nous souhaitons utiliser un système par classement. C'est-à-dire que chaque comportement exprime ses préférences sous la forme d'une liste ordonnée d'options. Plus particulièrement, nous avons décidé d'utiliser une des variations de la méthode de Borda qui attribue des points aux options en fonction de ce classement, l'ensemble des listes de préférences et des points étant utilisé ultérieurement pour dégager une préférence commune.

Dans cette même optique, les comportements classeront nécessairement toutes les options. Cela permet de ne pas tomber dans un défaut de la méthode qui est de ne pas classer des options pour défavoriser certaines au profit d'autres. Si un comportement transmet un classement incomplet, le mécanisme de sélection d'actions classera automatiquement dernières ex aequo les options n'ayant pas été classées, (et leur attribuera les points en conséquence.)

Les points affectés aux options en fonction de leur classement sont attribués comme dans l'exemple des points minimaux (cf. tableau III.1), c'est-à-dire comme suit : 0 point pour une première place, 1 point pour une seconde place, 2 points pour une troisième, *etc*.

Dans ce paragraphe nous montrons selon quels critères un comportement peut classer des propositions et donc comment les points sont répartis par le système de vote. Nous distinguons deux types de critères permettant de réaliser un classement soit en comparant les options entre elles, soit selon que les options vérifient ou non une condition.

3.6.1. Classement des options en fonction de préférences relatives

Un comportement est responsable de la manière dont il établit son classement. Celui-ci varie en fonction de la partie de l'application prise en charge. Pour établir son classement, un comportement doit utiliser un critère pour déterminer si une option doit être préférée à une autre. Nous proposons de donner quelques exemples :

Un comportement « inertie » tel que présenté dans la bibliographie favorise les options proches du choix précédent. Il établira donc son classement des options en fonction de leur distance avec le choix précédent. La distance entre deux options o_i et o_j de dimension n est donnée par la formule suivante :

$$d(o_i, o_j) = \sqrt[n]{\sum_{k=1}^{n} |o_{i,k} - o_{j,k}|^n} \quad \text{(formule III.6)}$$

Si v_{t-1} est le choix de l'agent à l'instant précédent, alors l'option o_i est préférée à l'option o_j si la distance entre o_i et v_{t-1} est inférieure à la distance entre o_j et v_{t-1}. Ce qui peut être résumé par la formule suivante :

$$d(o_i, v_{t-1}) < d(o_j, v_{t-1}) \Rightarrow o_i > o_j$$
avec $o_i > o_j$ signifiant o_i est préférée à o_j (formule III.7)

Un autre exemple simple et générique : un comportement est responsable de la régulation de l'une des variables de commande. C'est le cas de comportements tels que « maintenir une vitesse idéale » (cf. [Sukthankar, 1997]). Ce type de comportement compare les options selon une seule composante *k*, en fonction de la proximité de la composante des options avec une valeur objectif *obj*. L'option o_i est donc préférée à o_j si et seulement si la composante *k* de o_i est plus proche de *obj* que la composante *k* de o_j. Ce qui peut être noté :

$$|vo_{i,k} - obj| < |vo_{j,k} - obj| \Rightarrow o_i > o_j$$
avec $o_i > o_j$ *signifiant* o_i *est préférée à* o_j (formule III.8)

De même, dans le cadre de déplacements, il est possible de définir des critères de préférences à partir de critères géométriques quelconques. Dans le cadre général, un comportement peut classer les options selon qu'elles maximisent ou minimisent n'importe quelle fonction dépendante de ses composantes, d'un objectif de l'agent ou d'un objet de l'environnement.

Pour prendre en compte le fait qu'un comportement n'ait aucune préférence entre deux options, nous permettons aux comportements de classer des options ex aequo. Un comportement exprime donc ses préférences sous forme d'un pré-ordre total. Dans ce cas le nombre de points accordés aux options *ex aequo* diffère de ce que préconise la méthode de Borda.

3.6.2. Cas d'options *ex aequo*

La répartition des points entre les *ex aequo* est réalisée de la manière suivante : les options se partagent la somme des points correspondant aux places qu'elles occupent.

Par exemple, si les deux premières options sont *ex aequo,* elles se partagent les points de la première et seconde place soit 1 point (0+1). Le mécanisme de sélection d'actions leur attribue donc 0,5 point à chacune.

De même si trois options sont 4ème ex aequo, elles se partagent les points de la 4ème, 5ème et 6ème place soit 12 points au total ou 4 pour chacune. Les points attribués aux autres options ne changent pas : la 3ème récolte 2 points et la 7ème, 6 points.

Comme les comportements peuvent exprimer leurs préférences sous forme de pré-ordres, il leur est possible de classer les options selon qu'elles vérifient ou non un critère.

3.6.3. Classement des options en fonction de conditions binaires.

Le fait de pouvoir attribuer des points aux options selon qu'elles vérifient ou non une condition permet de ne pas établir de réel classement. En effet, il ne s'agit plus de classer les solutions entre elles mais simplement par rapport à une condition. Cette possibilité est bien

distincte de l'utilisation de veto. Le but des veto est d'exprimer un refus ; avec cette méthode, le comportement exprime une préférence non plus pour une mais pour un groupe d'options.

Les options doivent malgré tout être classées et des points leur sont distribués. Considérons n options dont m vérifient la condition. Si une option vérifie la condition, elle est classée première *ex aequo*. Si elle ne la vérifie pas elle est classée $(m+1)^{ème}$ *ex aequo*. Le pré-ordre suivant est donc établi :

$$o_1 \sim o_2 \sim \cdots \sim o_m > o_{m+1} \sim o_{m+2} \sim \cdots \sim o_n$$
où $o_i \sim o_j$ si $(non(o_i > o_j) \text{ et } non(o_j > o_i))$ (formule III.9)

Etant donné que les options *ex aequo* se partagent les points de manière équitable alors, les points sont attribués comme suit :

- Les m options qui vérifient la condition reçoivent ensemble la somme des entiers de 0 à m-1. Soit chacune $(m\text{-}1) / 2$ points[22].
- Les n-m options qui ne vérifient pas la condition reçoivent chacune $(n+m\text{-}1)/2$ points.

La répartition des points entre des options selon une condition est résumée dans le tableau ci-dessous (avec nécessairement $n \geq m$).

Tableau III.2 : *Répartition des points entre des options selon une condition, avec : n nombre total d'options à évaluer et m nombre d'options correctes.*

	Option	Classement	Points attribués
Options qui satisfont la condition	o_1	1ères *ex aequo*	$\frac{\sum_{i=0}^{m-1} i}{m} = \frac{m-1}{2}$
	o_2		
	...		
	o_m		
Options qui ne satisfont pas la condition	o_{m+1}	m+1ème *ex aequo*	$\frac{\sum_{i=m}^{n-1} i}{n-m} = \frac{n+m-1}{2}$
	o_{m+2}		
	...		
	o_n		

Dans l'exemple suivant, cinq options sont considérées. Quel que soit le classement, 10 points sont distribués (0+1+2+3+4). La répartition de ces 10 points est uniquement fonction du nombre d'options respectant la condition. Trois options vérifient la condition et se partagent (0 + 1 + 2) soit 3 points. Les deux options qui ne la vérifient pas se partagent 3+4 = 7 points.

22 $\frac{0+1+\ldots+(m-1)}{m} = \frac{m*(m-1)}{2*m} = \frac{m-1}{2}$

Tableau III.3 : *exemple de répartition des points entre des options selon une condition (avec m=3 et n=5).*

	Option	Classement	Points attribués
Options qui satisfont la condition	o_1	1ères *ex aequo*	$\frac{3-1}{2}=1$ *point*
	o_2		
	o_3		
Options qui ne satisfont pas la condition	o_4	4ème *ex aequo*	$\frac{3+5-1}{2}=3{,}5$ *points*
	o_5		

Cette étape permet donc aux comportements d'exprimer leurs préférences. Pour illustrer la manière dont les préférences sont établies nous avons expliqué comment nous souhaitons voir les points répartis entre les options. Rappelons que ces points sont attribués par le mécanisme de sélection d'actions (et non par les comportements eux-mêmes). Cela permet d'éviter les erreurs ou les « fraudes ».

L'étape suivante est la 6ème et dernière étape de la sélection d'actions. Elle est assez simple parce que la méthode de Borda a été utilisée.

3.7 Etape 6 : arbitrage final.

Les options valides ont été évaluées par l'ensemble des comportements. Conformément au principe de la méthode de Borda, des points ont été attribués en fonction de ces classements. Comme nous avons choisi cette méthode, le classement final des options est obtenu en réalisant la somme des points de chaque option. L'option totalisant le moins de points est sélectionnée.

Il est possible que plusieurs options aient obtenues le même nombre minimal de points. Dans ce cas elles doivent être considérées comme *ex aequo* mais il convient néanmoins d'en sélectionner une.

3.7.1. Options *ex aequo* à la fin de la procédure

Le problème de l'égalité des options (à la fin de la procédure) n'est pas mentionné dans la bibliographie traitant des architectures orientées comportement. Le vote par estimation moyenne qui est la méthode habituellement retenue est vraisemblablement moins sujette aux problèmes d'égalité. Comme nous le verrons dans le prochain chapitre, ce cas survient assez souvent dans notre application. Cela est dû à l'utilisation de points entiers[23], par rapport aux notes réelles attribuées par l'estimation moyenne : la probabilité étant plus faible puisqu'il y a plus de possibilités.

De nombreuses procédures de vote n'aboutissent pas nécessairement. Elle proposent alors une procédure permettant de déterminer un vainqueur. Nous pourrions également départager les options *ex aequo* selon un critère simple (et sans doute même générique). Cela ne nous est pas apparu primordial. En effet, si des options sont *ex aequo* à la fin de la procédure c'est que les comportements les ont jugées « équivalentes ». Comment le mécanisme de sélection d'actions pourrait-il les départager dans ce cas ? Le paradoxe de « Fredkin » cité dans [Minsky, 1988] nous rappelle que ce problème n'est sans doute pas si important qu'il y paraît : « Plus deux solutions paraissent aussi attirantes l'une que l'autre, plus il risque d'être difficile de choisir entre-elles, alors que dans ce cas, le choix importe d'autant moins. » En accord avec ce principe, les solutions égales sont départagées par le hasard.

Le fait que le hasard entre en compte dans la décision des agents peut avoir des répercussions sur le système multi-agents car il n'est plus possible de prédire le comportement d'un agent. De plus, les simulations dans lesquelles le hasard est intervenu ne peuvent plus être reproduites à l'identique.

23 Les points ne sont pas forcément entiers lorsqu'ils sont répartis entre des options ex aequo (selon le classement d'un comportement) mais dans ce cas ils sont identiques.

4. Conclusion

Dans ce chapitre nous sommes partis des inconvénients des architectures orientées comportement utilisant le vote et des objectifs fixés au chapitre II. Nous proposons une amélioration des architectures fondées sur le vote de préférences distribuées. Elle met en oeuvre des comportements de granularité réduite, utilise une méthode de vote autre que l'estimation moyenne et sélectionne des actions dans un domaine continu. Le tableau III.4 est un récapitulatif des caractéristiques de notre proposition comparées aux détails relatifs au vote de préférences distribuées et mis en avant dans la bibliographie (cf. tableau II.6, page 61). Les caractéristiques en caractères gras correspondent aux améliorations que nous avons réalisées ; les autres sont inspirées de la bibliographie sur les architectures orientées comportement.

Au niveau des comportements, les améliorations se traduisent par le fait qu'ils peuvent à la fois proposer des alternatives et les évaluer. Ces changements se répercutent sur les options et veto. De plus, la manière dont les comportements expriment leurs préférences est également modifiée : celles-ci sont exprimées par l'intermédiaire de classements des alternatives. Cela permet d'utiliser différentes méthodes de vote, nous avons retenu celle de Borda [Borda, 1781] (entre autres citée dans [Hudry, 2003]).

Afin de diminuer la granularité des comportements ceux-ci sont instanciés dans l'architecture en fonction du contexte environnemental ou des objectifs de l'agent. Ces modifications imposent de synchroniser les comportements pendant les différentes phases de la sélection d'actions. Cela nous conduit à définir un mécanisme fondé sur six étapes :

1. initialiser les comportements,
2. générer les options et veto,
3. spécifier les options,
4. relever les options valides,
5. évaluer les options,
6. établir les préférences globales.

Tableau III.4 : *récapitulatif des caractéristiques de notre proposition et des adaptations réalisées par rapport à l'état de l'art.*

Détail du modèle décisionnel		***Caractéristique de notre proposition***
Espace d'actions		**Continu**
Nombre de propositions par comportement		**Pas de limitation**
Priorités entre les comportements		**Non obligatoire**
Moyen d'assurer la persistance des choix		Comportement d'inertie
Granularité des comportements		**Très faible**
Nombre de comportements		**Variable**
Place de la planification		En dehors du système comportemental
Vote	Méthode	**Méthode par classement**
	Utilisation des veto	Oui

Le chapitre suivant est dédié à la validation du modèle dans le cadre de la navigation autonome en environnement virtuel. La validation nous permettra de montrer que le modèle est applicable à un problème concret, nous présenterons le développement incrémental du modèle décisionnel. Lors des tests réalisés, nous essaierons de quantifier les performances de

notre modèle et de souligner les phénomènes émergents (résultant des interactions entre comportements ou entre agents).

Chapitre IV : Validation du modèle proposé : application à la navigation autonome.

Dans le chapitre précédent nous avons exposé notre modèle décisionnel basé sur le vote de préférences distribuées. Notre proposition vise à améliorer les modèles initiaux ([Rosenblatt ,1996], [Tyrell, 1993], [Sukthankar, 1997] et [Hostetler et al., 2002]) en utilisant un espace d'actions continu, une granularité faible des comportements et en changeant la procédure de vote. Notre souhait sous-jacent est de diminuer l'influence des pondérations associées aux votes afin de réduire les problèmes de mise au point inhérents aux modèles orientés comportement. Il convient, dans ce présent chapitre de montrer comment ce modèle peut être appliqué à un problème concret. Nous évaluerons également le modèle afin de mesurer l'impact des modifications réalisées et d'en connaître les limites.

La première partie du chapitre présente l'application pour laquelle le modèle décisionnel est testé, à savoir la navigation en environnement simulé. Nous détaillerons en particulier l'environnement et l'agent pour lequel le modèle décisionnel est utilisé. La description de l'application nous permettra, de lister les critères que l'on souhaite vérifier et les essais devant être mis en oeuvre pour y parvenir.

La seconde partie de ce chapitre concerne la mise en oeuvre du modèle. Nous débuterons cette partie par l'implémentation du modèle général, puis nous détaillerons l'ensemble des comportements spécialement dédiés à la navigation autonome ou à l'évaluation du modèle. Pour chaque comportement nous définirons ses perceptions, les options et veto qu'il propose ainsi que la manière dont il exprime ses préférences lors du vote.

La troisième et dernière partie de ce chapitre présente les évaluations du modèle. Lors des expérimentations, nous commençons par comparer diverses implémentations de notre modèle soit directement entre elles, soit par rapport à d'autres modèles décisionnels proches comme l'AFAG (cf. chapitre II). Nous réalisons ensuite des tests en environnement multi-agents. Enfin, nous montrons les limites de notre modèle, en augmentant progressivement la densité des agents. Nous analyserons enfin les résultats obtenus durant les expérimentations.

Nous conclurons ce chapitre en rappelant les points positifs et négatifs de l'évaluation. A partir de ces remarques, nous envisagerons les améliorations à apporter au niveau de l'application ainsi qu'au niveau du modèle décisionnel.

1. Navigation autonome en environnement simulé

Nous avons choisi d'appliquer notre modèle à la navigation autonome. En effet, c'est une application courante, notamment dans le cadre des acteurs de synthèse mais qui pose encore de nombreux problèmes (y compris en simulation). De plus la navigation est l'application privilégiée du vote de préférences distribuées et c'est une application suffisamment dynamique pour justifier l'emploi d'une méthode réactive orientée comportement.

Dans cette partie nous aborderons la description des agents et de l'environnement que nous utiliserons pour notre validation. Nous rappellerons comment celle-ci a pu être réalisée dans la bibliographie et nous terminerons par les critères que nous souhaitons mesurer pour effectuer nos tests.

1.1 Agents

Les personnages virtuels, bien que possédant une représentation tridimensionnelle se déplacent généralement dans un plan (ou sur une surface pouvant y être localement assimilée). La gestion de leurs déplacements est donc un problème du plan. Pour ([Fruin, 1971] cité dans [Lamarche, 2003]) l'espace au sol occupé par un piéton est modélisé par un ovale de 50*60cm. Plus couramment et par soucis de simplification[24], les personnages sont approximés par des disques correspondant à l'empreinte au sol de leur cylindre englobant (cf. figure IV.1).

Figure IV.1 : *Approximation géométrique d'un personnage virtuel [Lamarche, 2003].*

Un agent est donc caractérisé par un rayon, une position. Le modèle décisionnel prenant en charge les déplacements d'un agent doit établir, en fonction des perceptions, une consigne permettant de déplacer le personnage.

La perception de nos agents est simple : l'agent perçoit tout objet présent dans un disque de rayon fixe, centré sur la position du personnage. L'agent est alors capable de percevoir les objets qui se trouvent derrière lui. Cette hypothèse simplificatrice est généralement prise en simulation de foule [Hostetler, 2003].

24 En particulier pour ne pas devoir tenir compte de leur orientation dans les calculs d'intersection.

Lier un modèle décisionnel avec un module d'animation s'avère particulièrement complexe (cf. chapitre I). Comme [Reynolds, 1999] nous admettrons qu'il existe une interface en ces deux modules offrant une solution correcte et permettant de s'affranchir de ce problème. Conformément à notre proposition en chapitre III, une consigne (et par conséquent une option) sera modélisée par un vecteur dont la première composante correspond à la direction de l'agent et la seconde à la vitesse. La fonction permettant de calculer l'évolution de l'agent en fonction de l'option o_i sélectionnée est notée en formule IV.1.

$$\theta A_{t+1} = o_{i,0}$$
$$vA_{t+1} = o_{i,1}$$
$$xA_{t+1}(o_i) = xA_t + o_{i,1}\cos(o_{i,0})$$
$$yA_{t+1}(o_i) = yA_t + o_{i,1}\sin(o_{i,0})$$

avec :
$\theta A_t \in [0, 2\pi[$: l'orientation de l'agent A à l'instant t
$vA_t \in [0, vA_{max}[$: la vitesse de l'agent A à l'instant t
xA_t : Abscisse de l'agent A à l'instant t
yA_t : Ordonnée de l'agent A à l'instant t

(formule IV.1)

1.2 Environnement

L'environnement des agents mobiles que nous venons de décrire est constitué d'obstacles statiques mais également de voies sur lesquelles ils évoluent. La modélisation de ces objets est abordée dans les paragraphes suivants.

Voies

Les voies sur lesquelles circulent les agents[25] dans notre environnement de test sont planes. Une voie est définie par son axiale (en pointillés sur la gauche de la figure IV.2) et sa largeur. Les voies sont modélisées par un enchaînement de segments et d'arcs de cercles créés à partir des points de l'axiale. Ce modèle « arcs-segments » [Thomas, 1999] permet de localiser les objets (et les agents) sur la voie (coordonnées curvilignes). Cela est possible car le positionnement possède une continuité à la fois pour l'abscisse et l'ordonnée curvilignes car les primitives de modélisation se joignent sans se chevaucher. De plus, la largeur constante de la voie permet de simplifier le modèle décisionnel en restant proche de la réalité. Le modèle de classes UML (« *Unified Modeling Language* ») d'une voie est reporté sur droite de la figure IV.2. Une voie est composée de tronçons qui peuvent être des virages ou des segments.

25 Les voies modélisées peuvent correspondre à des voies présentes dans la simulation mais peuvent également être utilisées pour définir un chemin emprunté par un agent (perspectives).

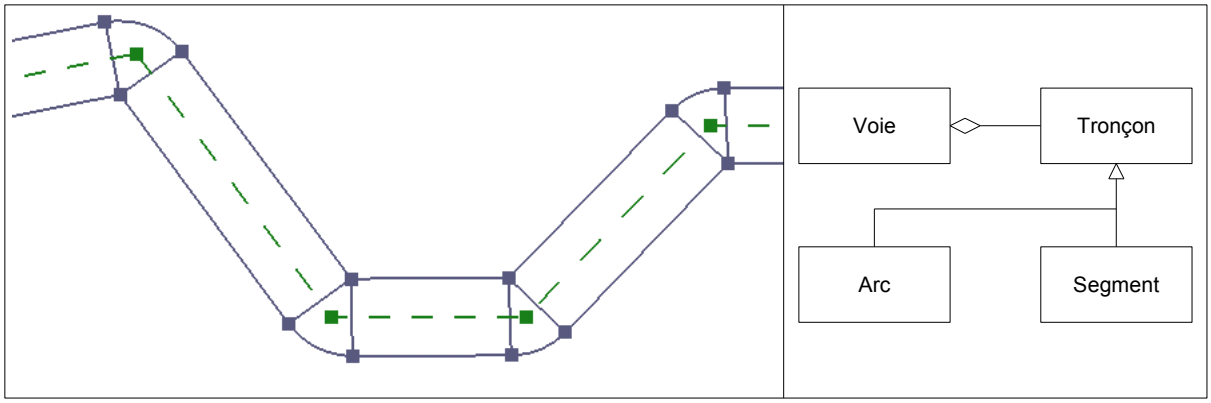

Figure IV.2 : *Modèle de Route, modèle segment-arc de cercle inspiré de [Thomas, 1999]*

Obstacles statiques

Les obstacles statiques sont positionnés par rapport à l'axiale de la route. Par souci de simplification (des fonctions d'intersection, de détection de collision, *etc.*) ils sont modélisés par des disques.

1.3 Évaluation des modèles de navigation autonome

L'évaluation d'un système, quel qu'il soit, ne prend son sens que lorsque les performances sont comparées avec d'autres modèles. Avant de donner les critères utilisés, nous nous permettons de souligner les paramètres qui influent sur un modèle décisionnel.

1.3.1. Paramètres influant sur le modèle décisionnel

Domaines d'application

Les modèles dont nous nous inspirons proviennent de différents domaines, en particulier la robotique mobile et l'informatique graphique. Les contraintes et objectifs inhérents à ces deux domaines étant fortement éloignés, il convient de nuancer les comparaisons. Notamment la différence entre simulation et application réelle fait encore débat. De plus, beaucoup de validations effectuées dans le cadre de l'informatique graphique sont subjectives. Enfin, le nombre d'agents considérés dans certaines simulations est impressionnant mais notons qu'au sein de ces masses d'agents les petits défauts ne sont pas toujours visibles.

Nombre d'agents

Dans le tableau IV.1 nous avons essayé de positionner le vote de préférences distribuées par rapport aux autres modèles de navigation pour acteurs virtuels. En haut du tableau se trouvent les applications mono-agent. Pour [Pettre, 2002] la navigation se base sur des algorithmes de planification sophistiqués mais applicables dans un environnement statique. En bas du tableau nous avons positionné le modèle présenté dans [Tecchia et al., 2000] qui simule des villes comportant jusqu'à 50 000 personnages. La méthode de navigation est basée sur des grilles et reste donc simple. Néanmoins les résultats peuvent être intéressants si l'on se place à une échelle permettant d'embrasser du regard un grand nombre d'agents. Le vote de préférences distribuées se situe dans les applications mono-agent et celles comportant moins de dix agents. Notre modèle qui dérive du vote de préférences distribuées permet actuellement de simuler jusqu'à 20 agents (avec de bons résultats), nous reviendrons sur ce point lors des essais.

Tableau IV.1 : *Comparaison du nombre d'agents simulés pour quelques modèles de navigation.*

Nombre d'agents	***Auteurs***	***Application***	***Méthode***
Un	[Pettre, 2002]	Monde virtuel désert	Algorithme de planification
	[Rosenblatt, 1996]	Robotique	Vote de préférences distribuées
Moins d'une dizaine	[Hostetler et al., 2002]	Piétons virtuels	Vote de préférences distribuées
	[Sukthankar, 1997]	Simulation de conduite sur autoroute	Vote de préférences distribuées
Plusieurs centaines	[Helbling et al., 2000]	Simulation de foule	Forces
	[Reynolds, 2000]	Nuées	Combinaison d'actions
Un millier	[Lamarche, 2003]	Piétons virtuels	Forces et algorithme dédié
Plusieurs milliers	[Tecchia et al., 2000]	Simulation de foule	Grilles

La densité des agents simulés n'est pas le seul critère de difficulté d'un environnement de test. La nature des relations entretenues entre les agents influe également sur la difficulté du problème à résoudre. Dans de nombreuses applications de navigation multi-agents (ou multi-acteurs, ou multi-robots) les agents appartiennent à un même groupe et n'ont pas d'objectifs contradictoires, ne se croisent pas, etc. Cette hypothèse forte contribue à simplifier le modèle décisionnel des agents.

1.3.2. Critères de performances du modèle de navigation

[Sukthankar, 1997] est le seul auteur proposant une évaluation objective de son modèle décisionnel basé sur le VPD. La fonction d'évaluation qu'il suggère (première colonne du tableau IV.2) retourne une valeur correspondant à l'agrégation de différents critères (cf. seconde colonne). Le score du modèle décisionnel est égal à la distance parcourue par l'agent à laquelle des points sont retranchés à chaque fois qu'une erreur est commise (collision, veto sur l'ensemble de l'espace d'actions, etc.). Prendre en compte la distance parcourue et vérifier que l'agent sort à la bonne sortie permet de valider que les objectifs de l'agent ont été réalisés. De même, un bon modèle décisionnel doit pouvoir anticiper les situations de blocage ; 10 000 points sont soustraits à chaque fois que le modèle décisionnel dépose des veto contre toutes les possibilités. Le modèle est également pénalisé à chaque collision car cela reflète son incapacité à respecter les contraintes. Enfin, le cumul des variations de vitesse et de position latérale permet de vérifier si l'agent est capable de persévérer dans ses choix.

Tableau IV.2 : *Evaluation du modèle de navigation par [Sukthankar, 1997].*

Évaluation dans le cadre de la navigation	***Qualités correspondantes du modèle décisionnel***
Évaluation =	
+ distance_parcourue	Réalisation d'un objectif
- 10000 × tout_veto	Anticipation du modèle
- 1000 × nb_collisions	Respect des contraintes
- 500 × mauvaise_sortie	Réalisation d'un objectif
- 0,02 × changement_de_vitesse	Persistance dans un choix
- 0,02 × déviation_latérale	Persistance dans un choix

La formule de Sukthankar met en avant des critères importants à vérifier pour la validation d'un modèle décisionnel de navigation. Néanmoins, il ne nous semble pas important d'agréger les évaluations. [Rosenblatt, 2000] propose d'évaluer la « sécurité » du chemin choisi en mesurant la distance moyenne avec l'obstacle le plus proche. Ce critère intéressant pour certaines applications ne nous semble pas toujours pertinent car le fait de longer un obstacle fixe permet de minimiser les risques de collision avec les obstacles mobiles.

La bibliographie concernant l'évaluation des modèles de navigation qui nous intéressent n'étant pas abondante, nous avons choisi de proposer nos propres critères.

Proposition de critères mesurables

Nous avons essayé de définir nos objectifs et critères de validation par rapport aux 9 critères de [Maes, 1989] et [Tyrrell, 1993] (cf. Chapitre II). Cela s'avère parfois difficile car ces critères sont davantage destinés à l'évaluation d'animats que de systèmes de navigation. De plus, certains critères mesurés peuvent être interprétés selon plusieurs des recommandations.

BUTS. Le modèle décisionnel d'un agent doit lui permettre de satisfaire des buts personnels. Concrètement, pour chaque expérimentation des modèles, il sera donc important de mesurer l'avancement des agents vers leurs objectifs. Nous définirons donc la notion de distance utile parcourue comme la différence d'ordonnée curviligne entre le point de départ et le point d'arrivée. Lorsque qu'un agent cherche à atteindre ses buts, il doit respecter certaines contraintes. Pour montrer que le modèle décisionnel respecte les contraintes nous relevons le nombre de sorties de route et ainsi que le nombre de collisions avec les obstacles ou les autres agents.

SITUÉ. Quelques modifications réalisées ont pour objectif de mieux lier l'agent avec son environnement, notamment l'utilisation d'un espace continu et l'instanciation dynamique des comportements. Il ne semble pas possible de mesurer si ce critère est respecté mais il est important de vérifier si ces modifications ont un impact positif sur les autres critères. Nous effectuerons une comparaison entre notre modèle et une implémentation discrète de celui-ci.

ANTICIPATION. Afin de vérifier si le modèle est capable d'anticiper les situations de blocage. Nous mesurerons comme [Sukthankar, 1997] le nombre de fois où aucune option n'est valide. Cependant, cet indicateur n'est pas suffisant pour détecter un blocage. Nous considérons également qu'il y a blocage lorsque l'agent n'a pas parcouru une distance utile suffisamment importante en un nombre de cycle donné. (Dans nos programmes de test un blocage est comptabilisé si l'agent a parcouru moins de 30 UD depuis les 80 cycles). La comparaison des distances utile et totale parcourues permet également de vérifier si l'agent anticipe ou non les obstacles.

PERSISTANCE. Ce critère est particulièrement important dans le cadre de la navigation autonome de piétons virtuels. Il convient de citer les travaux de [Goffman, 1971] (cité dans [Thomas, 1999] et [Lamarche, 2003]), qui mettent en évidence que les piétons ont tendance à minimiser leurs changements de direction. Nous relèverons donc la variation maximale et moyenne de direction.

OPPORTUNISTE ET COMPROMIS. Un agent exhibe un comportement opportuniste lorsqu'il décide d'accomplir une action permettant de réaliser un but secondaire facilement atteignable. De même, pour faire des compromis, l'agent doit posséder plusieurs buts. Ces critères ne peuvent donc être facilement validés dans le cadre de la navigation car un agent navigue vers un objectif unique.

RAPIDITÉ. Pour quantifier la rapidité du modèle décisionnel, nous proposons de mesurer le nombre moyen et maximal de propositions valides. Ces données étant des facteurs importants de la rapidité du mécanisme de sélection d'actions.

ROBUSTESSE. Comme nous l'avons déjà précisé, peu de modèles décisionnels peuvent être qualifiés de robustes. Pour tester la robustesse de notre modèle nous proposons d'adjoindre à l'ensemble des comportements, un comportement « parasite » dont les propositions et les évaluations relèvent du hasard. Les variations de performance par rapport à un modèle sain permettent de mesurer la robustesse du modèle.

RÉUTILISATION. La possibilité de réutiliser un modèle décisionnel est un critère subjectif difficilement mesurable. Il convient de vérifier que l'ajout de nouveaux comportements n'est pas trop complexe (construction incrémentale). De même il doit être possible d'étendre le modèle. (Nous aborderons ce point dans les perspectives.)

Le tableau IV.3 ci dessous donne un récapitulatif des critères que nous venons de présenter.

Tableau IV.3 : *récapitulatif des critères mesurables lors de la validation.*

Qualité	***Comportements observables***	***Critères mesurables***
Buts	Avance, ne reste pas bloqué Respecte les contraintes	Distance parcourue (totale et utile) Sortie de route (Nombre et ordonnée curviligne maximale) Nombre de collisions
Situé	Réalise des choix adaptés à la situation présente	Non mesurable directement possède des répercussions sur les autres critères. Comparaison avec modèle discret.
Anticipation	Ne pas devoir revenir en arrière Anticipe les obstacles.	Rapport distance utile / parcourue Nombre de blocages individuels Nombre de cycles sans solution
Persistance	Trajectoire rectiligne	Variations de direction (moyen et maximal)
Opportunisme	Difficilement mesurables dans l'application présente car l'agent ne possède pas plusieurs buts distincts	
Compromis		
Rapidité	Ne ralentit pas l'animation	Nombre d'options valides et invalides par cycle (moyen et maximal) Nombre de comportements (moyen et maximal)
Robustesse	Comportement parasite	Performance du modèle dégradé
Réutilisation	Facilité de mise en oeuvre et possibilité d'enrichissement du modèle	Critère subjectif qui n'est pas directement mesurable, la partie implémentation donne un exemple d'application du modèle

Les critères utiles à la validation de notre modèle ayant été définis, nous montrons dans la partie suivante son implémentation dans le cadre de la navigation.

2. Application du modèle proposé à la navigation

La navigation constitue une application intéressante permettant de valider la majorité des aspects du modèle décisionnel que nous proposons. Avant d'exposer les tests de validations proprement dits, nous proposons de présenter les détails concernant l'implémentation du modèle dans le cadre de la navigation réactive. Nous estimons, en effet, qu'implémenter un modèle afin de l'appliquer à un cas particulier n'est pas trivial car beaucoup d'hypothèses et de choix sont faits durant ce processus.

Cette partie de chapitre est construite selon l'ordre chronologique dans lequel l'implémentation a lieu. Nous commençons par décrire la conception du coeur de l'agent. Nous essayons, ensuite, de spécifier l'ensemble des comportements qui doivent être développés. Enfin, et conformément au principe de construction incrémentale, nous développons les comportements en les intégrant au fur et à mesure dans le modèle décisionnel des agents.

2.1 Implémentation du « coeur » des agents

2.1.1. Architecture générale d'un agent

Les agents ont été conçus selon une approche orientée objet. Les classes utilisées sont présentées sur le diagramme de classes UML en figure IV.3. En suivant une cohérence et pour éviter l'existence de dépendances cycliques, nous avons scindé nos agents en deux principales classes, à savoir, « M*odèle physique* » et son héritière « *Noyau décisionnel* ». La classe « M*odèle physique* » comporte les membres qui décrivent la partie « opérative » de l'agent (taille, position, orientation, vitesse ...) ainsi que la mémoire des options et veto considérés. Cette classe définit également les méthodes permettant à l'agent d'agir (de se déplacer) en fonction de l'option sélectionnée. La classe « Noyau décisionnel » porte les comportements et définit les méthodes responsables de la perception et de la cognition de l'agent.

La classe « *Comportement* » dépend des classes « *Modèle physique* », « *Option* » et « *Veto* ». Cette classe abstraite est étendue pour construire les classes correspondant aux comportements utilisés par l'agent.

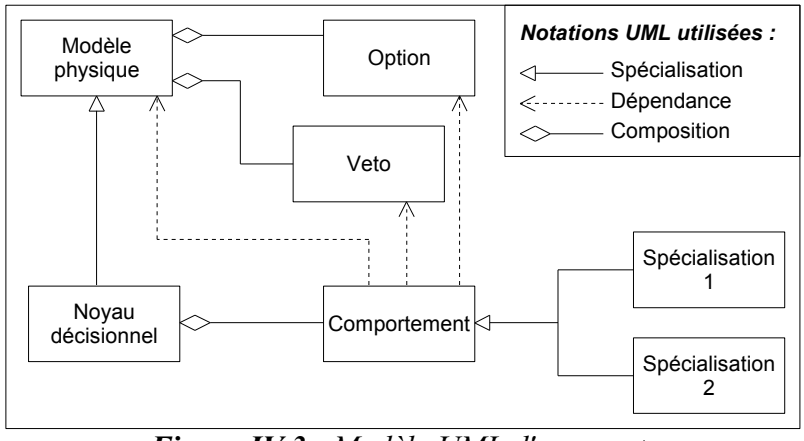

Figure IV.3 : *Modèle UML d'un agent.*

La classe *comportement* implémente rigoureusement le modèle de comportement proposé au chapitre III. Nous avons ajouté dans les classes *option* et *veto* des membres permettant de les décrire.

2.1.2. Étiquetage des options et veto

Les architectures orientées comportement sont souvent très difficiles à mettre au point, en particulier parce que les données numériques qu'elles manipulent ne peuvent pas être appréhendées facilement. De plus, ces informations sont perdues après la fusion ou après l'arbitrage. Pour aider à la compréhension des options et des choix de l'agent nous proposons « d'étiqueter » les options et les veto.

Le concepteur d'une application choisit les informations contenues dans la désignation des options. Nous pensons qu'il est utile d'indiquer le type et l'instance du comportement qui a proposé l'option. Si ce comportement peut proposer plusieurs alternatives, alors il n'est pas inutile de mentionner à quelle alternative l'option correspond. De cette manière, les désignations associées aux options peuvent servir au concepteur en l'aidant à comprendre le sens des choix (données numériques) de l'agent. Cela s'avère important pour valider le bon fonctionnement d'un agent ou, au contraire, diagnostiquer une erreur. Comme nous le montrerons par la suite, ces informations supplémentaires peuvent également être utilisées par le modèle décisionnel lui-même.

A titre d'exemple, une option proposée par un comportement d'évitement statique attaché à un objet particulier correspondant à un évitement par la gauche peut être notée comme suit dans le tableau IV.4.

Tableau IV.4 : *Exemple de désignation d'une option.*

Type de comportement	***Instance de l'objet concerné*** [26]	***Alternative***
Éviter obstacle statique	123	Passer à gauche

Remarque : Précisons, que les dénominations ne sont pas attachées aux options mais à leurs composantes. En effet, après l'exécution de l'étape 3, « *spécifier les options brutes »,* il est possible que la même option soit composée de composantes provenant de diverses options et de divers comportements. C'est pour cela que la description d'une option doit décrire chacune de ses composantes.

Nous venons de décrire les classes qui composent notre agent, nous avons également proposé quelques adaptations afin de faciliter l'application de notre modèle. Nous allons exposer la manière dont l'algorithme de sélection d'actions est implémenté au sein du cycle global d'un agent.

26 Dans un environnement simulé, les objets peuvent être référencés par une clef unique, l'ajout de ce type d'information est à rapprocher de la notion « d'affordance » (cf. chapitre I). Dans un environnement réel il serait difficile de différencier les instances d'une même classe d'objets.

2.1.3. Cycle global d'un agent

Notre mécanisme de sélection d'actions s'inscrit dans le cycle perception, cognition, action d'un agent selon l'algorithme ci-dessous.

Algorithme IV.1 : *Cycle perception, cognition, action d'un agent.*

```
//Perception
    Oublier les options et veto du cycle précédent
    (étape 1) Instancier les comportements
//Décision
    (étape 2) Recevoir les options et veto de chaque comportement
    (étape 3) spécifier les options brutes
    (étape 4) relever les options valides
    Si il existe au moins une solution valide alors
        (étape 5) Évaluer les options valides
        (étape 6) Sélectionner la meilleure option
    Sinon
        Choisir de rester sur place
    fin si
//Action
    Déplacer le personnage en fonction de l'option considérée
```

Afin d'appliquer notre modèle, il faut concevoir les comportements qui seront utilisés et implémenter la fonction qui les instancie (étape 1 de l'algorithme). Les comportements sont conçus comme des spécialisations de la classe « *comportement* »

2.1.4. Identification des comportements

L'une des difficultés de l'approche orientée comportement réside dans la division du problème en un ensemble cohérent de comportements complémentaires. Notons que ce problème est récurrent en informatique, qu'il s'agisse d'identifier des agents ou de simples sous-programmes. Les méthodologies ne proposent que peu de directives précises pour cette étape essentielle. Bien que ne disposant pas d'une méthodologie précise, nous essaierons de suivre les grands principes de l'approche orientée comportement :

- «Encapsulation », au sein d'un comportement, des données et des méthodes relatives à ce qui est pris en compte par ce comportement (perception, décision et action).
- Mise en oeuvre de comportements simples, réactifs lorsque cela est possible, conformément au « Principe de parcimonie » [Drogoul, 1993] qui conseille de toujours vérifier qu'un agent possède bien la granularité minimale nécessaire.
- Construction incrémentale des agents.

Comme pour [Hostetler, 2003], nous débuterons nos développements par un comportement « *suivre la voie* » qui choisit l'orientation du personnage et un comportement « *maintenir la vitesse* » qui décide de la vitesse. Nous développerons ensuite le comportement permettant de prendre en compte les obstacles statiques, puis le comportement « *inertie* » destiné à la persistance des choix. Nous développerons le comportement « *éviter un obstacle*» et terminerons par le comportement « *éviter un agent* » (qui est conçu comme une spécialisation de l'évitement d'un obstacle statique). La figure IV.4 ci-dessous illustre la manière dont la classe abstraite « *comportement* » est redéfinie afin de créer le modèle décisionnel complet de l'agent. Le comportement « *parasite* » est utilisé pour les validations.

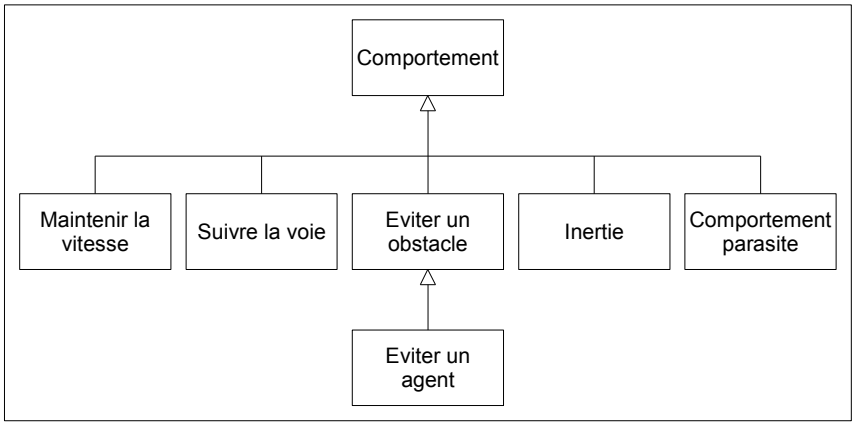

Figure IV.4 : *Spécialisation de la classe « comportement » (Diagramme de classe UML).*

Nous venons d'établir la liste des comportements qui seront utilisés par le modèle décisionnel. Nous commençons par développer le comportement permettant à l'agent d'avancer sur une voie libre.

2.2 Comportement « suivre la voie »

Le développement de notre modèle de navigation débute par la mise en oeuvre du comportement « *suivre la voie* ». Nous ne détaillerons pas les différentes solutions étudiées lors de la conception de ce comportement. Nos choix ont été guidés par un souci de simplicité comparable au principe de parcimonie de [Drogoul, 1993]. Afin de définir ce comportement, il convient d'écrire la liste des données perçues, les options et veto susceptibles d'être proposés et enfin les critères selon lesquels sont évaluées les alternatives.

Les données perçues par ce comportement sont les suivantes :

- Données relatives à l'agent :
    - Rayon de l'agent
    - Sens de déplacement sur la voie
    - Ordonnée curviligne de l'agent
- Données relatives à la voie :
    - Direction de la tangente à l'axe de la route
    - Largeur de la voie

L'orientation de la voie au point où l'agent est positionné (tangente à l'axiale) ainsi que le sens dans lequel il se déplace permettent de calculer une orientation privilégiée. La largeur de la voie, le décalage de l'agent par rapport à l'axiale (ordonnée curviligne) et le rayon approximant l'agent permettent de vérifier que l'agent ne sorte de la voie.

Par conséquent, le calcul des options et veto s'avère extrêmement direct et ne dépend d'aucun état interne au comportement. Les informations perçues conduisent à proposer :

- lorsque l'agent est éloigné de la limite de la voie (cf. figure IV.5), une option dont la direction est celle de la tangente à la voie.
- lorsque l'agent est proche de la limite de la voie (cf. figure IV.6), un veto qui interdit à l'agent de sortir de sa voie et une option qui permet de l'écarter de la bordure.[27]

27 L'intervalle du veto est un angle de 180° centré sur la normale à la voie. L'option permettant de recentrer l'agent correspond à la tangente à l'axiale ± 11.5° (0,2 radian).

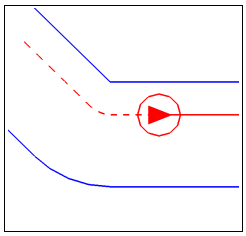

Figure IV.5 : *Proposition du comportement « Avancer dans sa voie » lorsque l'agent est éloigné de la limite de la voie*

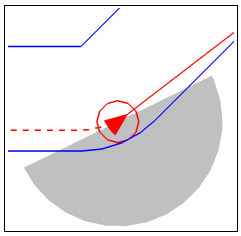

Figure IV.6 : *Proposition et veto du comportement « Avancer dans sa voie » lorsque l'agent est proche de la limite de la voie*

Afin de distribuer la sélection d'actions entre les comportements, nous avons souhaité que ceux-ci puissent, en plus de proposer des options et déposer des veto, évaluer l'ensemble des options. Ce comportement classe les solutions en fonction de leur proximité avec sa propre proposition. Ce critère a été formulé, dans le chapitre III (formules III.6 et III.7)

La vitesse de l'agent est constante et réglée par un comportement simple qui propose une option dont l'orientation est indéfinie et la vitesse égale à une constante. Tant que l'environnement est statique ce comportement n'influe pas sur la sélection d'actions.

A ce stade du développement, l'agent est donc capable de se déplacer sur une voie dégagée de tout obstacle avec la garantie de ne jamais s'en écarter.

2.3 Comportement «éviter un obstacle»

Le comportement développé précédemment permet à l'agent d'évoluer sur une voie libre. Dans cette seconde étape du développement de notre agent, nous proposons donc de créer un nouveau comportement lui permettant d'éviter les obstacles. Comme nous l'avons souligné dans le chapitre III, nous attachons les comportements à des aspects très spécifiques de l'environnement. Nous développons, par conséquent, un comportement permettant d'éviter un unique obstacle, ce comportement pouvant être instancié plusieurs fois. Contrairement à la description du comportement précédant, nous avons souhaité mettre en avant plusieurs des possibilités envisagées pour concevoir le comportement.

2.3.1. Options et veto proposés par le comportement « éviter un obstacle »

Options proposées

Nous n'avons retenu qu'une seule possibilité pour les propositions faites par le comportement « *éviter un obstacle* ». Par souci de simplification, et comme le montre la figure IV.7, nous considérons que les obstacles sont de forme circulaire ou qu'ils peuvent être approximés comme tel. Les options proposées par le comportement « *éviter un obstacle* » sont les trajectoires tangentes à cet obstacle passant par le centre de l'agent. Un idée similaire est retenue dans [Lamarche, 2003].

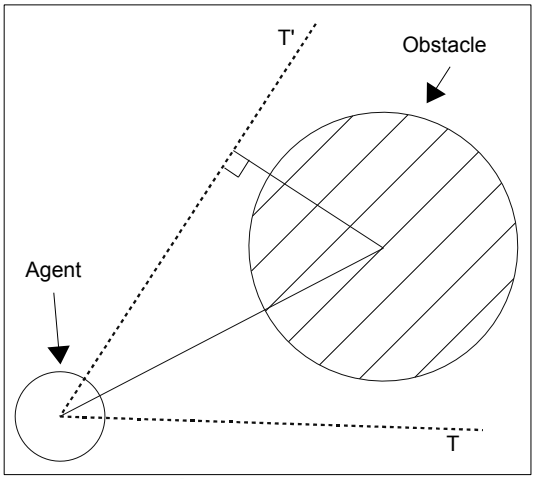

Figure IV.7 : *Évitement d'obstacle statique.*

La vitesse n'ayant pas d'influence sur l'évitement d'un obstacle statique (dans notre application), les options retenues o_1 et o_2 sont donc notées :

$o_1=(T,*)$
$o_2=(T',*)$
avec : T et T' l'angle entre les tangentes à l'obstacle et l'axe des abcisses

(formule IV.2)

Les options non pertinentes peuvent avoir un effet important sur le vote. En effet, elles augmentent inutilement le nombre d'options évaluées, ce qui allonge le temps nécessaire à l'exécution du modèle. De plus, elles peuvent influencer le résultat du vote, même si elles n'ont aucune chance d'être sélectionnées. C'est pourquoi les options décrites sur la formule IV.2 ne sont proposées que lorsque l'obstacle se trouve devant l'agent.

Les deux options de ce comportement offrent une alternative pour contourner l'obstacle. Elles ne permettent pas d'empêcher la collision. Ceci est précisément le rôle des veto dont les descriptions suivent.

Veto proposés

Les veto sont utilisés pour éviter que l'agent ne percute l'obstacle. Nous proposons deux possibilités pour ce veto, que nous avons qualifiées de veto « faible » et de veto « fort ». Le veto fort interdit toute action allant dans la direction de l'obstacle. Le veto faible interdit toute action qui engendrerait une collision au pas suivant de la simulation.

Le veto fort interdit toute option qui, quelle que soit sa vitesse, se dirige vers l'obstacle. Il est donc défini par le secteur angulaire compris entre les deux tangentes T et T' sur la figure IV.7 et par une vitesse quelconque. Le veto est donc noté comme suit.

$v_1=(]T;T'[,*)$
avec : T et T' l'angle entre les tangentes à l'obstacle et l'axe des abcisses

(formule IV.3)

L'avantage de cette solution est qu'elle oblige l'agent à anticiper l'obstacle. Néanmoins, dans certains cas cette solution peut sembler abusive car elle invalide de nombreuses options. La figure IV.8 illustre cette situation. L'agent 1 dépose un veto contre la sortie de route et un veto fort pour chaque obstacle. Ces trop nombreux veto invalident des options pertinentes, et l'agent se trouve bloqué. C'est pour cette raison que nous proposons un veto moins contraignant. (Ce veto est utilisé par l'agent 2 sur la figure IV.8).

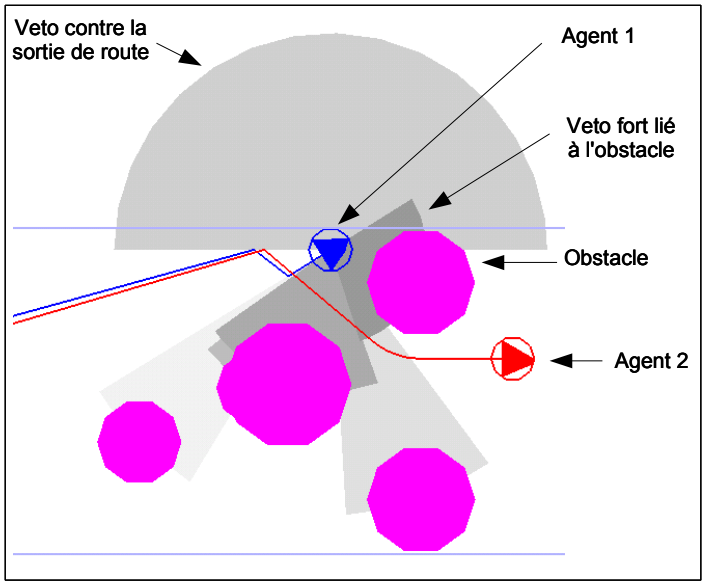

Figure IV.8 : *Blocage dû à l'utilisation abusive de veto.*

Le veto faible est défini comme suit. Compte tenu de la vitesse maximale de l'agent et de sa distance avec l'obstacle, il est possible de déterminer s'il peut y avoir collision. Si oui, un veto (cf. formule IV.4) est déposé afin de limiter la vitesse dans le secteur angulaire défini par les deux tangentes T et T' (cf figure IV.7).

$v_1 =]T ; T'[, [vitesse_1, vitesse_{Max}])$

$vitesse_1 = AO - r_a - r_o$

avec :

AO la distance agent-obstacle

r_a et r_o les rayons de l'agent et de l'obstacle.

(Les vitesses sont exprimées en UD par cycle)

T et T' l'angle entre les tangentes à l'obstacle et l'axe des abcisses

(formule IV.4)

Les veto permettent d'éviter les collisions. Les préférences de ce comportement sont exprimées lors du vote et doivent permettre d'anticiper les obstacles.

2.3.2. Votes de l'évitement d'obstacle

Le vote du comportement « *éviter un obstacle* » doit permettre d'anticiper son contournement en défavorisant les options qui dirigent l'agent vers cet obstacle. C'est pourquoi, dans le cas où l'on utilise un veto fort, nous ne jugeons pas nécessaire de développer cette fonction. Dans le cas contraire, celle-ci s'avère utile et nous proposons trois possibilités. La première possibilité n'exploite que les informations relatives à l'obstacle. La seconde, prend en considération le positionnement relatif de l'agent et de l'obstacle par rapport à la voie. La troisième solution fait intervenir une mémorisation au sein du comportement.

Vote sans prise en compte du reste de l'environnement

L'intérêt de construire un comportement sans faire d'hypothèse sur le reste du problème est de favoriser sa réutilisation dans d'autres contextes. Si le comportement ne considère que l'agent et l'obstacle, alors le comportement ne peut évaluer les options que sur un unique critère : selon qu'elles pointent ou non vers l'obstacle. Cette possibilité de classement est formulée dans le chapitre III. Le classement des options s'établit comme suit :

- Les options qui ne se dirigent pas vers l'obstacle sont classées 1$^{\text{ères}}$ *ex aequo*.
- Les options qui dirigent l'agent vers l'obstacle sont classée n$^{\text{ème}}$ *ex aequo*.

La conditions à la base du classement peut se noter comme suit :

$$rang(o_i):\begin{cases}=1 \text{ si } o_{i,1} \notin]T, T'[\\ >1 \text{ sinon}\end{cases}$$

avec : $rang(o_i)$ le rang de l'option i dans le pré-ordre des préférences du comportement
T et T' l'angle entre les tangentes à l'obstacle et l'axe des abcisses

(formule IV.5)

Nous attribuerons donc les points en suivant le tableau III.2 (chapitre III, page 86). Nous proposons une seconde solution qui exploite les informations relatives à l'environnement. Nous comparerons les deux méthodes par la suite.

Vote en prenant en compte le reste de l'environnement

Comme cela est préconisé dans [Thomas, 1999], il est judicieux « d'informer » l'environnement et d'exploiter ces informations afin de simplifier ou d'améliorer le modèle décisionnel des agents. Dans la solution précédente, nous n'exploitions pas ces informations or comme l'illustre la figure IV.9, il est possible d'utiliser les abscisses et ordonnées curvilignes de l'agent et de l'obstacle afin de déterminer si l'obstacle peut ou non gêner l'agent.

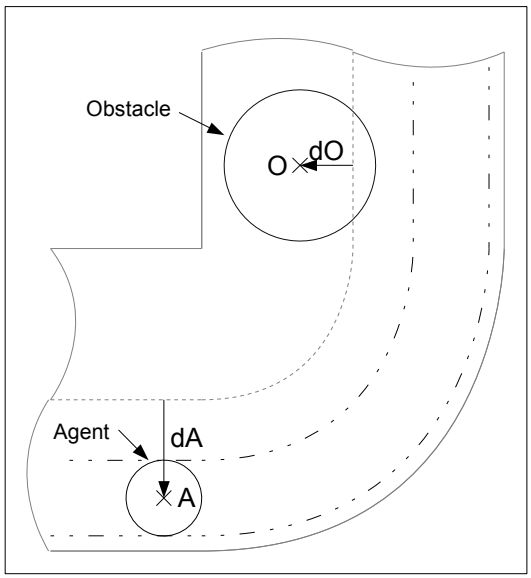

Figure IV.9 : *Évitement d'obstacle en utilisant les ordonnées curvilignes*

Les préférences du comportement peuvent être définies comme suit. Si l'obstacle n'est pas susceptible de gêner l'agent, alors le comportement ne vote pas. Si, en revanche, l'obstacle le gêne, alors le comportement favorise les options permettant de se décaler au maximum de

l'obstacle. Ce critère est défini par la formule IV.6 ci-dessous. (La formule IV.1 définit le calcul de la position future de l'agent en fonction d'une option.)

$|dA_{t+1}(o_i) - dO| > |dA_{t+1}(o_j) - dO| \Rightarrow o_i < o_j$

où :

$dA_{t+1}(o_k)$: l'ordonnée curviligne de l'agent à l'instant $t+1$ si l'option o_k est sélectionnée

dO : l'ordonnée curviligne de l'obstacle

$o_i < o_j$ se lit l'option o_i est préférée à l'option o_j

(formule IV.6)

L'avantage de cette solution est qu'elle encourage l'agent à continuer d'éviter l'obstacle du côté vers lequel il a commencé à dévier. Ce vote présente pourtant un défaut majeur, illustré par la figure IV.10. L'agent (au centre de la figure) essaie de contourner l'obstacle par la droite mais il se heurte à la limite de la route. La seule option valide (contourner à gauche) sera donc suivie. Or, au cycle suivant, les deux options de contournement sont à nouveau valides et l'option contourner à droite est sélectionnée. Au troisième cycle, l'agent est donc revenu dans sa position d'origine et ne quittera jamais ces deux états.

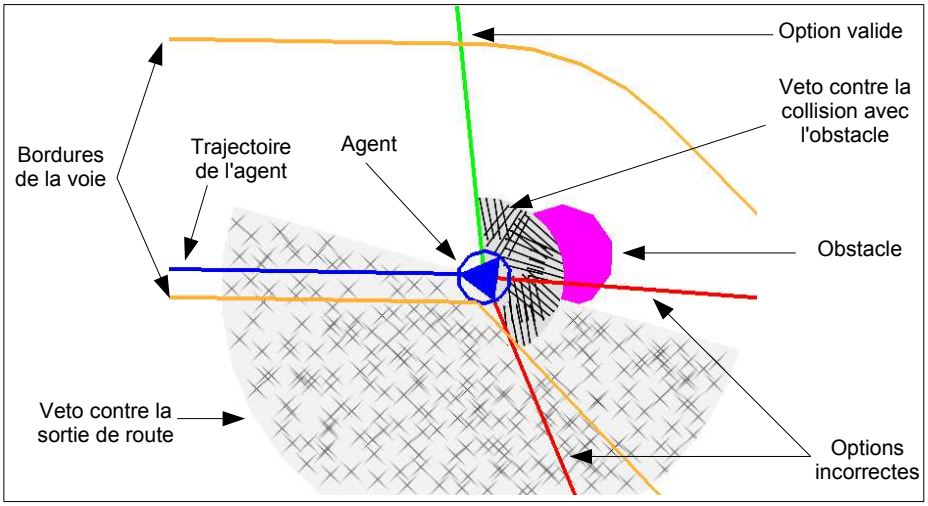

Figure IV.10 : *Situation de blocage dans l'évitement d'un obstacle.*

Pour ne pas rester continuellement dans cet état de blocage nous proposons d'introduire une mémorisation au sein du comportement.

Mémorisation des blocages et échanges d'informations entre les comportements

Lorsqu'un blocage a été détecté, il convient de modifier le vote du comportement d'évitement pour l'obliger à s'écarter du point où le blocage survient. L'apparition d'un blocage peut être repérée en utilisant les désignations des veto : l'un des veto correspondant à la sortie de route et l'autre à la collision avec l'obstacle. La figure IV.11 ci-dessous représente l'automate correspondant qui est mis en oeuvre dans le comportement.

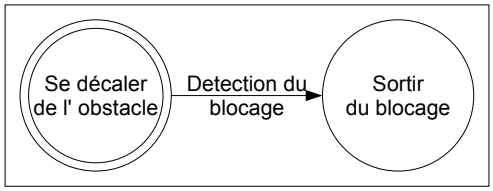

Figure IV.11 : *Ajout d'une mémoire interne au comportement*

Dans la pratique, l'utilisation de ce comportement n'est pas encore pleinement satisfaisante. Le tableau IV.5 présente l'ensemble des votes obtenus quand l'agent se trouve dans le cycle suivant celui qui est présenté en figure IV.10. Les deux solutions valides étant *ex aequo,* la sélection est aléatoire. Il apparaît donc un problème lié au choix aléatoire des options *ex aequo* : cycle après cycle, l'agent a peu de chances de contourner l'obstacle.

Tableau IV.5 : *votes de comportements lors de l'évitement d'un obstacle, défaut relatif au choix aléatoire des options ex aequo.*

Option	Veto	Points de « Suivre la route »	Points de «éviter l'obstacle »	Totaux Points	Classement final
Avancer dans la voie	Veto obstacle				
Contournement de l'obstacle à droite	-	0	1	1	1er *ex aequo*
Contournement de l'obstacle à gauche	-	1	0	1	1er *ex aequo*

Afin de résoudre ce problème d'hésitation lié à l'apparition de deux options *ex aequo* dans les votes, nous introduisons dans le modèle un comportement d'inertie qui permettra à l'agent de persévérer dans ses choix.

2.4 Comportement « inertie »

Faire appel au hasard lors de la sélection d'actions s'avère parfois pénalisant car l'hésitation qui en découle ne permet pas de sortir l'agent d'un « deadlock ». Nous mettons donc en place le comportement d'inertie permettant de persévérer dans un choix. Les détails de l'implémentation du comportement « *inertie* » ont été donnés dans le chapitre III. Dans l'exemple suivant, nous retenons un comportement d'inertie ne proposant pas d'option mais votant en faveur des options proches de l'option retenue au cycle précédent.

La figure IV.12 montre le déroulement de l'évitement d'obstacle en deux temps. La vignette de gauche présente l'instant où le blocage est détecté. La vignette de droite illustre la trajectoire suivie par la suite.

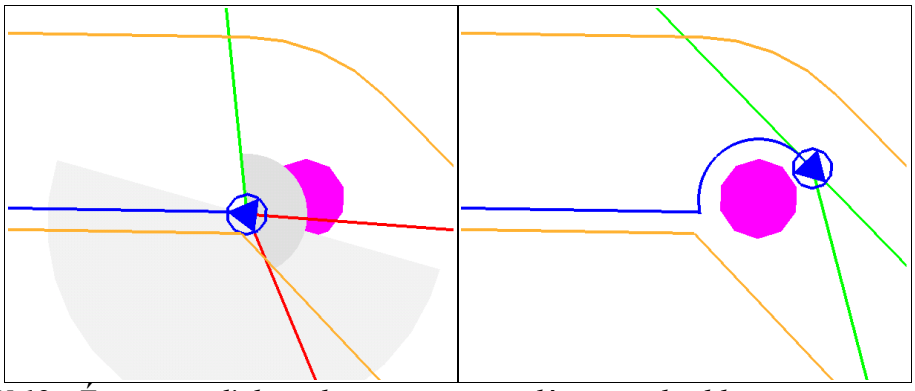

Figure IV.12 : *Évitement d'obstacle statique avec détection des blocages et comportement d'inertie.*

Lors du premier cycle, (partie gauche de la figure IV.12) une seule option est valide, l'agent n'a donc pas d'autre choix que d'éviter l'obstacle par la gauche. Lors des cycles suivants, deux options sont valides et conduisent à éviter l'obstacle vers la gauche ou vers la droite. L'agent peut, soit continuer à contourner l'obstacle par la gauche soit revenir vers la droite. Les votes de chaque comportement sont exprimés par le tableau IV.6. Le comportement d'inertie ainsi que celui évitement d'obstacle classent l'option contournant l'obstacle vers la gauche en premier. Par conséquent, cette dernière est retenue.

Tableau IV.6 : *Évitement d'obstacle, votes de comportements après la détection d'un blocage.*

Option	Veto	Points de « Suivre la route »	Points de « éviter un obstacle »	Points de l'inertie	Points totaux	Classement final
Avancer dans la voie	Veto obstacle	-	-	-	-	-
Contournement de l'obstacle à droite	-	0	1	1	2	2ème
Contournement de l'obstacle à gauche	-	1	0	0	1	1er e

A ce stade du développement, le modèle décisionnel permet à un agent de naviguer dans une voie encombrée d'obstacles statiques. Ce modèle décisionnel est utilisé dans le point quatre pour réaliser une comparaison avec un modèle à fusion d'actions généralisée [Arnaud, 2000] dans le cadre d'un environnement statique. Afin de répondre à notre problème de navigation dans un environnement dynamique, nous développons donc la prise en compte des autres agents dans les paragraphes suivants.

2.5 Comportement d'évitement inter-agents

Concevoir un agent capable de prendre en compte un environnement dynamique et continu, et d'interagir avec d'autres agents, constitue un exercice délicat. C'est pourquoi il s'avère nécessaire de faire un certain nombre d'hypothèses. Rappelons que nous avons choisi de dupliquer les comportements et que, par conséquent, à une instance de comportement est associée un unique agent à éviter (de même qu'une instance de comportement est associée à un unique obstacle statique). Nous avons également fait l'hypothèse que les agents en interaction se trouvent sur la même voie. Ces deux hypothèses permettent de limiter le nombre d'interactions entre agents et de les caractériser simplement.

Le comportement que nous avons développé possède une granularité légèrement plus importante que les comportements déjà présentés car il gère un aspect complexe de la navigation. Afin de prendre en compte les différentes alternatives auxquelles sont confrontés les agents en interaction, un automate est mis en oeuvre. En outre, cet automate permet également de modéliser l'aspect séquentiel de certaines interactions.

2.5.1. Diagramme d'état du comportement d'évitement inter-agents

La figure IV.13 représente l'automate modélisant le comportement d'évitement inter-agents (Par souci de simplicité, une partie des transitions a été omise). Lors de l'initialisation du comportement les transitions C0 à C3 permettent de sélectionner si l'agent évité doit être suivi ou doublé, s'il convient de ne rien faire ou s'il s'agit d'un croisement. L'action de doubler comporte 3 étapes qui permettent à l'agent de déboîter en accélérant puis de maintenir une vitesse rapide et enfin, lorsque l'action est terminée, de reprendre sa vitesse de préférence.

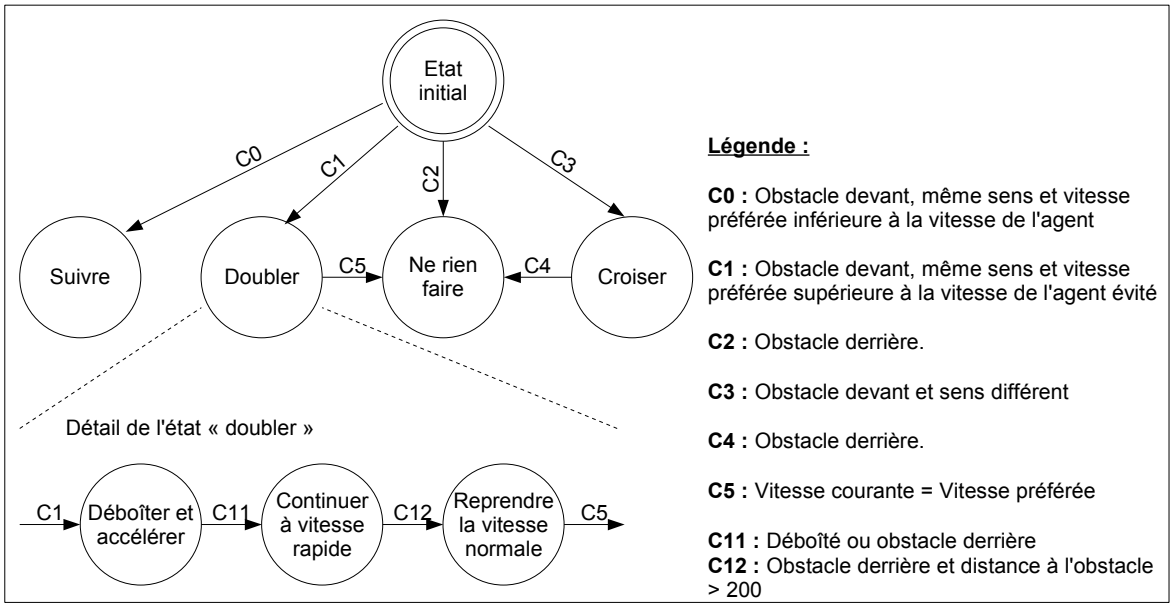

Figure IV.13 : *Diagramme d'états du comportement d'évitement inter-agents.*

Le comportement étant décrit par un automate à état fini, il convient de définir les options et veto proposés en fonction de l'état dans lequel se trouve l'automate.

2.5.2. Options et veto proposés par l'évitement inter-agents

La figure IV.14 présente les différentes possibilités pour éviter un obstacle mobile. Pour prendre en compte l'incertitude relative à la position future de l'agent évité, il est nécessaire de définir une zone de sécurité dont le rayon est défini par la vitesse maximale de cet agent.

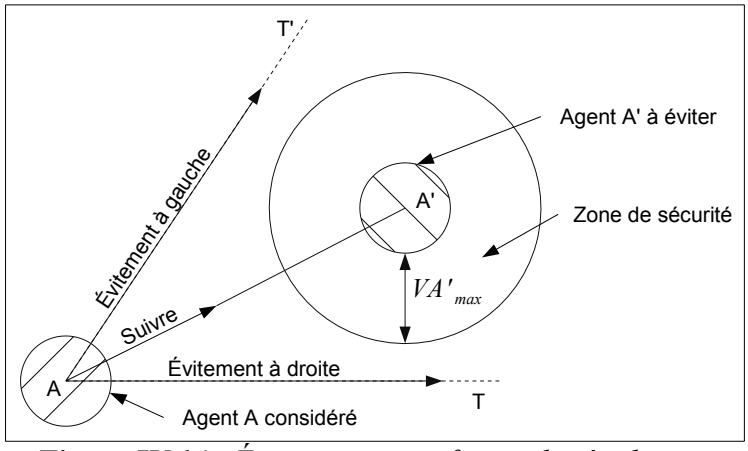

Figure IV.14 : *Évitement par « forces de répulsion »*

Dans un premier temps nous avions envisagé de ne pas tenir compte de l'état de l'automate et de proposer toutes les options quel que soit son état. Nous avons abandonné cette idée car cela conduisait à prendre en compte beaucoup de propositions inutiles (en particulier lorsque beaucoup d'autres agents étaient pris en compte par le modèle décisionnel). La formule IV.7 liste les différentes options proposées en fonction de l'état de l'automate de la figure IV.13.

Etat "croiser" :	Etat "déboiter" :
$o_{11}=(T,*)$	$o_{31}=(T,VA_{max})$
$o_{12}=(T',*)$	$o_{32}=(T',VA_{max})$
Etat "suivre" :	Etat "vite" :
$o_{21}=(\widehat{AA'},VA'_t)$	$o_{41}=(*,VA_{max})$

avec :
T et T' l'angle entre les tangentes à l'obstacle et l'axe des abcisses
$\widehat{AA'}$ l'angle entre l'axe (AA') et l'axe des abcisses
VA'_t la vitesse de l'agent A' à l'instant t
VA_{max} la vitesse maximale de l'agent A

(formule IV.7)

Le veto éventuellement déposé par ce comportement est similaire à celui de l'évitement statique (cf. formule IV.4) et permet d'éviter la collision à l'instant suivant de la simulation. L'unique différence étant qu'il faille également tenir compte de l'incertitude relative à la position future de l'agent évité. Pour prendre en compte cette incertitude, la zone prise en compte est une zone circulaire dont le rayon correspond à la vitesse maximale de l'agent.

Les veto et options relatifs à l'évitement inter-agents ayant été décrites, il reste à définir la la manière dont ce comportement exprime ses préférences.

2.5.3. Vote du comportement d'évitement d'agent

De même que pour les options proposées, le vote du comportement dépend de l'état dans lequel se trouve l'automate. Les différents critères de classement relatifs à chacun des états du comportement sont résumés par la formule IV.8. Comme pour l'évitement d'obstacle statique, nous proposons 2 alternatives pour le vote relatif à l'état « croiser » du comportement, nous verrons les avantages de chaque solution un peu plus loin. La fonction d'évaluation permettant de « déboîter » maximise le décalage et la vitesse. Pendant que l'agent double un autre agent la

vitesse est maximisée. Enfin, lorsque l'agent a terminé de doubler, le comportement favorise les options proches de la vitesse favorite. Les autres états n'évaluent pas les options.

Etat "croiser" :
$|dA_{t+1}(o_i) - dA'_t| > |dA_{t+1}(o_j) - dA'_t| \Rightarrow o_i < o_j$

ou
$rang(o_i)$: $\begin{cases} =1 \text{ si } o_{i,1} \notin]T, T'[\\ >1 \text{ sinon} \end{cases}$

Etat "déboiter" :
$(|dA_{t+1}(o_i) - dA'_t| + o_{i,1}) > (|dA_{t+1}(o_j) - dA'_t| + o_{j,1}) \Rightarrow o_i < o_j$

Etat "vite" :
$o_{i,1} > o_{j,1} \Rightarrow o_i < o_j$

Etat "Revenir à la vitesse normale"
$|o_{i,1} - VA_{pref}| < |o_{j,1} - VA_{pref}| \Rightarrow o_i < o_j$

avec :
$rang(o_i)$ le rang de l'option i dans le pré-ordre des préférences du comportement
$dA_{t+1}(o_k)$: l'ordonnée curviligne de l'agent à l'instant $t+1$ si l'option o_k est sélectionnée
dA'_t : l'ordonnée curviligne de l'agent A' à l'instant t
VA_{pref} la vitesse préférée de l'agent A
T et T' l'angle entre les tangentes à l'obstacle et l'axe des abcisses

(formule IV.8)

A ce stade du développement, l'agent est capable de naviguer dans un environnement dynamique si la densité n'est pas trop importante. Nous reviendrons sur ce point lors de l'évaluation du modèle. Avant de clore cette parie destinée à la mise en oeuvre du modèle, il reste à définir le comportement aléatoire qui sera utilisé pour évaluer la robustesse du modèle.

2.6 Comportement « parasite »

Le comportement « *parasite* » est utilisé pour montrer la robustesse de l'architecture. Il peut également sortir l'agent d'une situation de blocage (situation dans laquelle aucune option n'est valable, ou les options sont toutes défavorables). Le comportement parasite que nous avons développé propose une option et vote aléatoirement, il ne propose pas de veto.

Nous venons de décrire les détails concernant notre modèle de navigation. En particulier nous avons développé les comportements suivants : « *maintenir la vitesse*», « *suivre la voie* », « *éviter un obstacle* », « *inertie* » et « *éviter un agent* ». Nous proposons à présent d'évaluer les résultats de notre modèle. Il est d'autant plus important de l'évaluer que de nombreuses solutions sont envisageables pour chaque comportement.

3. Expérimentations

Lors des expérimentations nous comparons diverses implémentations de notre modèle, soit directement entre elles, soit par rapport à d'autres modèles décisionnels proches. Nous avons proposé deux solutions pour l'évaluation des options par le comportement permettant d'éviter un obstacle statique. Nous comparerons donc ces deux possibilités. Ensuite, nous montrerons les avantages de notre modèle continu. Nous montrerons également que notre modèle donne de meilleurs résultats que l'AFAG (cf. chapitre II). Nous utiliserons le comportement parasite afin de souligner la robustesse du modèle décisionnel. Enfin, nous testerons le modèle dans un environnement multi-agents. Nous terminerons cette partie par une discussion des résultats obtenus.

3.1 Comparaison des deux comportements d'évitement d'un obstacle statique

Nous avons proposé deux solutions pour le vote du comportement permettant d'éviter un obstacle statique. L'une est exprimée par la formule IV.5 et classe les solutions en deux catégories selon qu'elle se dirigent ou non vers l'obstacle. L'autre solution est exprimée par la formule IV.6 et classe les options en fonction du décalage avec l'obstacle. *A priori,* ces préférences s'avèrent équivalentes et c'est pourquoi nous proposons de les comparer.

Afin de comparer objectivement les deux solutions, nous avons réalisé une série de 50 simulations de 1 000 cycles chacune. Les agents évoluent sur une section de voie droite (afin de pouvoir comparer les distances utiles parcourues). Trois obstacles de rayon variables compris entre 30 et 60 UD, sont repositionnés aléatoirement à chaque exécution entre les abscisses curvilignes 1000 et 1200 et les ordonnées -50 et 50 de la voie.

Au début de la simulation, les deux agents mettant en oeuvre les deux comportements à comparer sont positionnés à l'origine de la route. La vitesse des agents est constante (2UD /cycle). La figure IV.15 ci-dessous montre un exemple des trajectoires suivies par chaque modèle, la plus rectiligne étant relative au comportement « bon / mauvais » (formule IV.5) l'autre au décalage (formule IV.6).

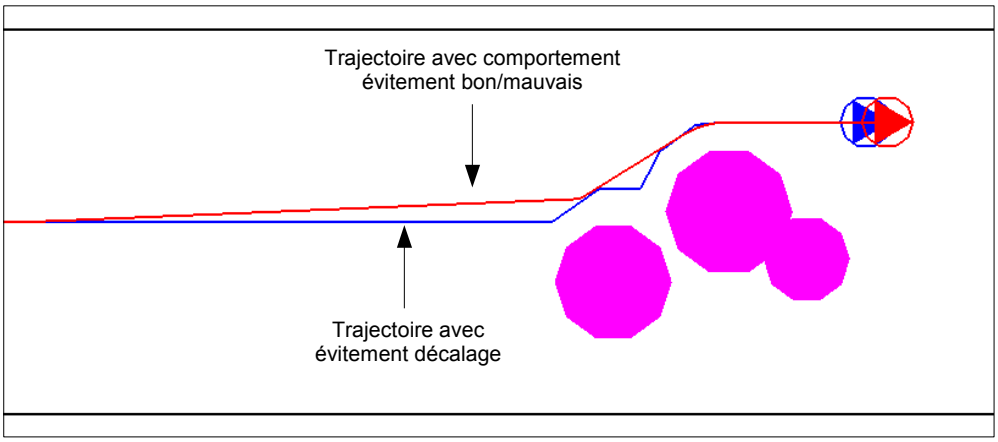

Figure IV.15 : *Trajectoires comparées des deux comportements d'évitement d'obstacle statique.*

Le tableau IV.7 ci-dessous résume les mesures réalisées, conformément aux critères présentés dans la première section de ce chapitre. Pour chaque mesure, nous indiquons les

valeurs moyennes, maximales et minimales relevées lors des 50 exécutions. Les valeurs minimales et maximales sont importantes car elles soulignent souvent les cas les plus défavorables qui devront être améliorés.

La distance totale parcourue est toujours de 2000 UD (1000 cycles, 2UD / cycle) car il n'y a pas eu de blocage. La distance utile parcourue varie en fonction des obstacles et des trajectoires suivies pour les éviter. Le modèle avec évaluation « bon / mauvais » (formule IV.5) parcourt plus de distance utile que le modèle « décalage » (formule IV.6). Le modèle « bon / mauvais » anticipe donc mieux les obstacles.

Le nombre de sorties de route et de collisions est toujours nul : le modèle respecte les contraintes. L'ordonnée curviligne maximale quantifie le dépassement de la contrainte liée à la route ; elle n'a donc pas d'importance dans le cas présent.

La variation moyenne de direction illustre la capacité de l'agent à persévérer dans ses choix. La variation maximale durant une simulation est importante car, pour une application graphique, un mouvement brusque d'un personnage n'est pas réaliste. Ici les choix peuvent varier au maximum de 125° pour le modèle « bon / mauvais » et de 180° pour le modèle « décalage ». Les chiffres montrent cependant que ces variations brusques sont assez marginales.

Le nombre de blocages souligne les simulations durant lesquelles le modèle s'est trouvé face à un problème qu'il ne savait pas résoudre. Le modèle « bon / mauvais » n'a jamais été bloqué. Le modèle « décalage » a été considéré bloqué pendant, au maximum, 471 cycles.

Le nombre de cycles sans solution est nul : dans aucun cas les agents n'ont invalidé toutes les options. Ce cas ne pouvait se produire dans ces environnements statiques pour lesquels il existait nécessairement un passage.

Les nombres d'options et de comportements donnent un aperçu de la rapidité d'exécution du modèle. Ici, le nombre de comportements est constant (trois comportements d'évitement d'obstacle, un comportement inertie et un comportement suivre la voie). Le nombre d'options est constamment inférieur ou égale à huit. (Il faut donc noter que dans cet environnent simple le nombre d'options est inférieur à celui utilisé dans les modèles similaires [Hostetler et al., 2003].)

Tableau IV.7 : *Résultats du modèle avec évaluation type bon / mauvais.*

	Modèle avec évaluation bon / mauvais.			***Modèle avec évaluation décalage***		
	Moy	***Max***	***Min***	***Moy***	***Max***	***Min***
Distance totale parcourue	2000	2000	2000	2000	2000	2000
Distance utile parcourue	1988	1999	1933	1929	1999	936
Sorties de route	0	0	0	0	0	0
Ordonnée curviligne maximale	78	121	27	76	122	27
Collisions	0	0	0	0	0	0
Variation de direction moyenne (degré)	0,04	0,37	0,01	0,8	18,47	0,01
Variation de direction maximale (degré)	18,49	125,98	2,94	61,36	180	6,01
Blocages	0	0	0	16,75	471	0
Cycles sans solution	0	0	0	0	0	0
Nombre moyen d'options valides (par cycle)	2,69	2,81	2,54	2,68	4,2	2,51
Nombre maximal d'options valides (par cycle)	8	8	8	7,98	8	7
Nombre moyen d'options invalides (par cycle)	0,09	0,35	0,01	0,23	1,91	0,02
Nombre maximal d'options invalides (par cycle)	4,71	6	2	5,47	6	3
Nombre moyen de comportements (par cycle)	3,64	3,71	3,6	3,69	4,32	3,6
Nombre maximal de comportements (par cycle)	5	5	5	5	5	5

Les mesures réalisées montrent que le modèle « bon/mauvais » donne de meilleurs résultats. En particulier, parce qu'il anticipe mieux les obstacles et persévère plus dans ses choix (la variation de direction est plus faible). Il faut souligner que « le modèle à décalage » s'est mis dans des situations de blocage (renforçant ainsi les meilleurs résultats de l'autre modèle). La figure IV.16 illustre une situation de blocage du modèle « à décalage ». Comme les obstacles se trouvent de part et d'autre de l'agent, ils provoquent des votes opposés qui s'annulent et laissent l'agent continuer tout droit. Pour le modèle décisionnel utilisant la formule IV.5, les trois comportements « bon/mauvais » ont tous tendance à défavoriser l'option du comportement « suivre la route ». Cette dernière n'est donc pas retenue et c'est une des options d'évitement qui remporte le vote.

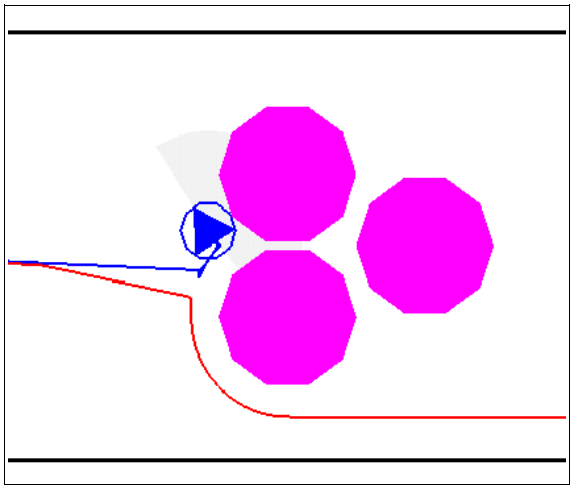

Figure IV.16 : *Situation de blocage du modèle « à décalage ».*

Les meilleurs résultats du modèle « bon / mauvais » (formule IV.5) s'expliquent donc simplement. Au delà de ce résultat applicatif, cet exemple met en avant l'un des problèmes inhérents aux architectures orientées comportement. En essayant de réaliser un compromis entre plusieurs options contradictoires, la méthode de coordination peut sélectionner une action qui ne satisfait pas la majorité des comportements. Dans le cas du vote selon le décalage avec l'obstacle, deux comportements (voie et inertie) sur les cinq (trois comportements d'évitement) sont « satisfaits » par le vote. Par rapport à ce qu'affirme la bibliographie, nous remarquons donc que le vote permet de résoudre ce problème de compromis uniquement pour certains cas particuliers. Pour le reste des implémentations, nous utiliserons donc le comportement d'évitement statique établissant ses préférences en classant les options en deux catégories (formule IV.5).

3.2 Comparaison espace d'action discret / continu

Nous avons mis ce test en oeuvre afin de vérifier que les résultats sont meilleurs en utilisant un espace d'action continu. Afin d'effectuer cette comparaison de manière objective, nous n'avons pas ré-implémenté le modèle de [Hostetler et al., 2002] dont les détails ne sont pas tous connus. Nous avons donc réalisé quelques adaptations à notre propre modèle décisionnel pour qu'il effectue ses choix dans un espace d'actions discret.

Pour l'adaptation discrète du modèle, nous avons modifié le programme pour que les options proposées par les comportements soient remplacées par cinq options fixées par avance. Ces cinq propositions sont calculées par rapport à l'orientation θ du mobile. Elles ont pour composante de direction : θ, θ ± 0,5 radian et θ ± 1.6 radian (soit respectivement 28,65° et 91,67°). Ces cinq options sont évaluées par les comportements et sélectionnées selon le principe exposé au chapitre III.

La figure IV.17 illustre la trajectoire du modèle discret et celle du modèle continu sur un circuit ne comportant aucun obstacle. Comme le modèle discret ne propose pas systématiquement d'option appropriée, l'agent est obligé de « louvoyer » autour de l'axe de la voie afin d'avancer.

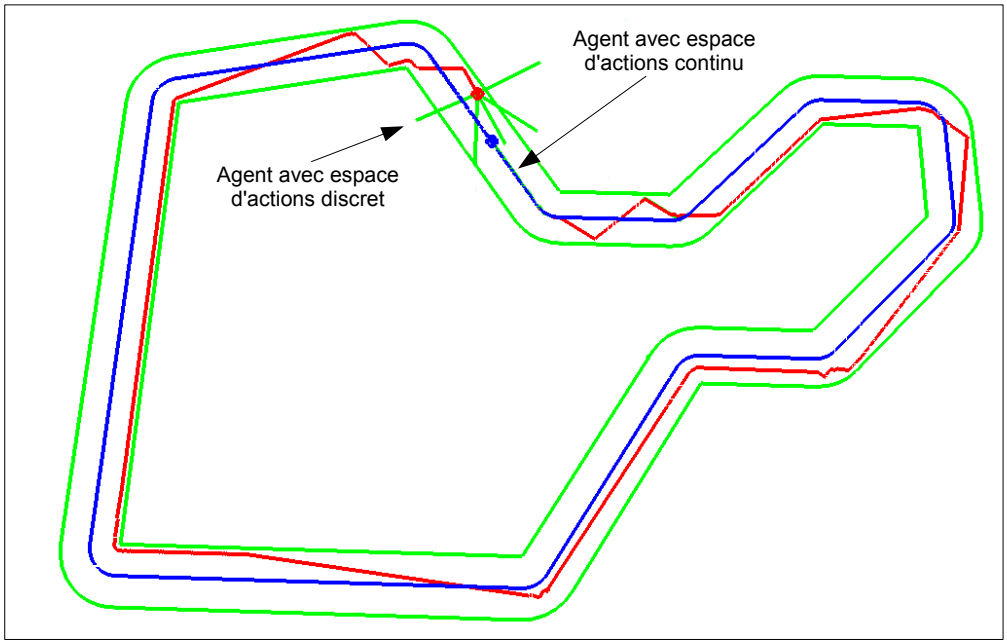

Figure IV.17 : *Trajectoires générées par le modèle discret et le modèle continu.*

La figure IV.18 est un exemple de trajectoire générée par le modèle discret sur un circuit comportant quelques obstacles. Comme pour l'exemple précédent, la trajectoire n'est pas souple. En outre, l'agent n'est pas capable de sortir de son blocage, car aucune option ne convient.

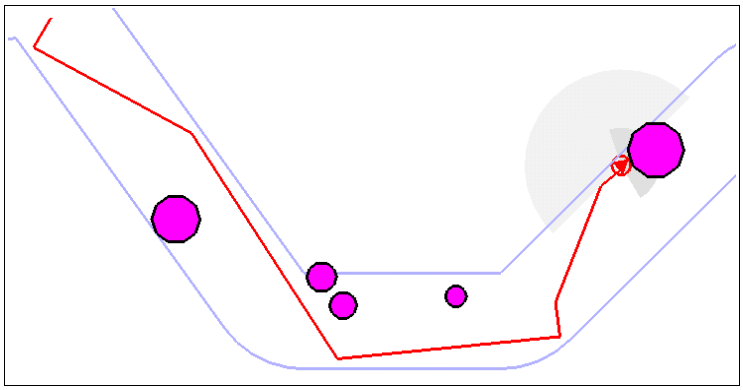

Figure IV.18 : *Trajectoire générée par le modèle discret.*

Les trajectoires suivies par les agents (cf. figures IV.17 et IV.18) soulignent le problème du modèle discret, à savoir qu'un choix acceptable ne peut être fait lorsqu'il n'y a aucune option acceptable dans l'espace d'actions. Notons que notre implémentation du suivi de voie renforce les mauvais résultats du modèle discret. En effet, contrairement aux autres comportements similaires de la bibliographie [Hostetler et al., 2002] [Reynolds, 1999] nous n'avons pas fait l'hypothèse que l'agent possède une ordonnée curviligne privilégiée. Les trajectoires du modèle discret seraient apparues correctes sur les figures en « plan large », mais les hésitations de l'agent auraient été très visibles sur une vue détaillée, lors de la simulation.

Ayant choisi le comportement d'évitement d'obstacle le plus favorable et montré l'avantage du modèle continu, nous pouvons comparer les résultats du modèle que nous proposons avec une autre architecture orientée comportement : l'Architecture à Fusion d'Actions Généralisée (AFAG) [Arnaud, 2000].

3.3 Comparaison avec un modèle utilisant l'AFAG

Nous avons essayé d'appliquer l'Architecture à Fusion d'Actions Généralisée [Arnaud, 2000] au problème de navigation de piétons virtuels ([Hanon et al., 2002], [Ebel et al., 2002] et [Hanon et al., 2003]). Il nous a donc semblé important de comparer les résultats de l'AFAG avec ceux de notre modèle décisionnel.

L'environnement dans lequel les essais ont été réalisés est le circuit présenté sur la figure IV.19. Les agents se déplacent à vitesse constante (2 UD par cycle) sur ce circuit. Cette voie est encombrée de 40 obstacles circulaires. Nous avons réalisé 50 simulations de 10 000 pas de temps, ce qui permet aux agents de réaliser un peu plus de deux tours du circuit pendant une simulation. L'agent utilisant une architecture AFAG est celui présenté dans le chapitre II. L'agent utilisant notre modèle met en oeuvre les comportements « *suivre la voie* », « *inertie* », et le comportement « *éviter un obstacle* » qui a été retenu précédemment (formule IV.5).

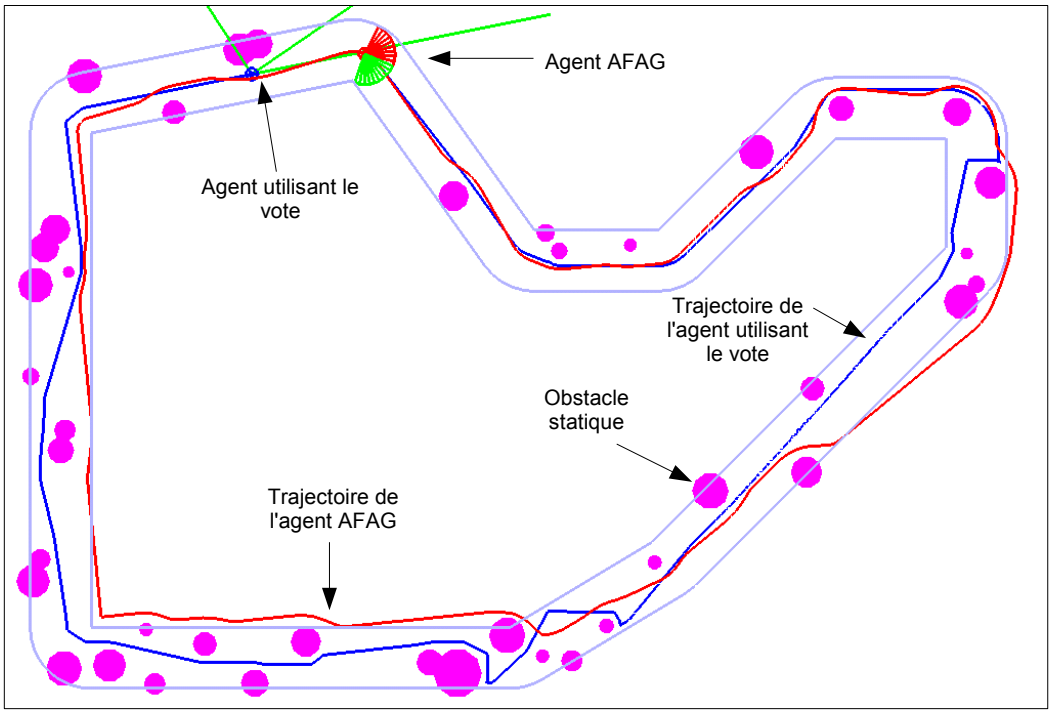

Figure IV.19 : *Trajectoires comparées d'un modèle utilisant l'AFAG et d'un modèle utilisant notre amélioration du Vote de Préférences Distribuées.*

Le tableau IV.8 récapitule les résultats des deux modèles (les résultats de l'AFAG sont constants c'est pourquoi nous ne donnons pas les valeurs moyennes, maximales et minimales). Les résultats des deux modèles sont similaires. La distance utile parcourue par l'AFAG est très légèrement supérieure car l'agent AFAG n'est pas toujours resté sur la voie. De même, les variations moyennes de direction sont très légèrement plus faibles pour notre modèle. La différence majeure entre les deux modèles réside dans le respect ou non des contraintes. Notre modèle n'entre en collision avec aucun des obstacles alors que l'agent AFAG entre en collision pendant 103 cycles de la simulation. De plus, l'agent AFAG sort ostensiblement de la route alors que notre modèle ne sort que légèrement et très ponctuellement. Ces sorties de route de notre agent s'expliquent par un défaut dans la formulation du veto : il devrait être déposé avant que l'agent n'arrive en bordure de la route. Ce changement minime peut être réalisé simplement.

Tableau IV.8 : *Résultats comparés de l'AFAG et de notre modèle. (Les colonnes grisées sont sans objet).*

	Vote de Préférences Distribuées			***AFAG***
	Moy	***Max***	***Min***	
Distance totale parcourue	20000	20000	20000	20000
Distance utile parcourue	18995	19174	18845	19835
Sorties de route	52,96	66	35	692
Ordonnée curviligne maximale	81,74	81,78	81,73	99,96
Collisions	0	0	0	103
Variation de direction moyenne (degré)	0,42	0,48	0,38	0,64
Variation de direction maximale (degré)	180	180	180	90
Blocages	0	0	0	0
Cycles sans solution	0	0	0	
Nombre moyen d'options valides (par cycle)	6,01	6,07	5,95	
Nombre maximal d'options valides (par cycle)	14	14	14	
Nombre moyen d'options invalides (par cycle)	0,38	0,43	0,33	
Nombre maximal d'options invalides (par cycle)	11	11	11	
Nombre moyen de comportements (par cycle)	7,99	8,01	7,97	
Nombre maximal de comportements (par cycle)	12	12	12	

Il faut également noter que, si les variations moyennes de direction sont inférieures, les variation maximales sont plus importantes pour notre modèle. Cependant, les fortes variations de direction correspondent le plus souvent à un rebroussement de chemin suite à la détection d'un mauvais choix (cf. figure IV.20) et sont donc justifiables.

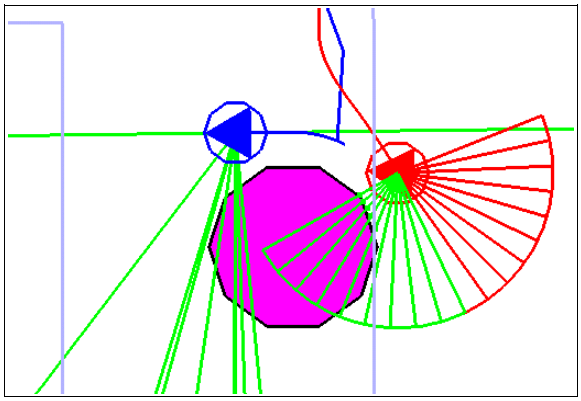

Figure IV.20 : *Changement brusque de trajectoire après blocage.*

Les tests effectués par rapport à l'AFAG montrent que si les résultats sont comparables quant aux distances parcourues et aux variations moyennes de direction, notre modèle présente l'avantage majeur de respecter les contraintes inhérentes à la tâche de navigation.

Bien qu'il soit difficile de montrer la robustesse d'un modèle décisionnel nous proposons une évaluation du modèle dans un fonctionnement dégradé.

3.4 Fonctionnement dégradé par un comportement parasite

Pour réaliser les tests du modèle décisionnel en mode dégradé, nous avons repris le même environnement que celui du test précédent (figure IV.19). Le comportement parasite présenté précédemment a été ajouté à l'agent. Le tableau IV.9 donne les résultats du modèle. Il apparaît que les résultats du modèle restent du même ordre, seule la persistance des choix n'est plus assurée (la variation moyenne de direction passe de 0,42° à 2,97°).

Tableau IV.9 : *Résultats de notre modèle en mode dégradé.*

	Moyen	***Max***	***Min***
Distance totale parcourue	20000	20000	20000
Distance utile parcourue	19351,04	19643,8	18943,7
Sorties de route	8,06	29	2
Ordonnée curviligne maximale	81,34	81,88	80,61
Collisions	0	0	0
Variation de direction moyenne (degré)	2,97	3,47	2,65
Variation de direction maximale (degré)	180	180	180
Blocages	32,36	148	0
Cycles sans solution	0	0	0
Nombre moyen d'options valides (par cycle)	6,84	6,89	6,76
Nombre maximal d'options valides (par cycle)	15	15	15
Nombre moyen d'options invalides (par cycle)	0,46	0,52	0,38
Nombre maximal d'options invalides (par cycle)	10	10	10
Nombre moyen de comportements (par cycle)	8,94	8,96	8,93
Nombre maximal de comportements (par cycle)	13	13	13

L'adjonction d'un comportement parasite dans le modèle ne dégrade donc pas excessivement les résultats. Ce test ne prouve pas la robustesse du système dans tous les cas. En particulier, il ne donne pas d'indication sur ses performances lorsqu'un comportement important fonctionne dans un mode dégradé. (Ce cas peut se produire lorsque le modèle décisionnel est physiquement distribué.) Néanmoins, l'intérêt de ce résultat et de montrer que le système est ouvert car il tolère l'ajout d'un comportement quelconque. Cette propriété est importante car elle assure le concepteur d'une application qu'il peut facilement étendre son modèle.

Nous venons de mettre en avant quelques avantages de notre modèle décisionnel dans des environnements statiques. Nous allons, à présent, présenter les résultats en environnement dynamique.

3.5 Validation du modèle pour des interactions multi-agents

Le modèle de navigation réactive que nous proposons, fonctionne correctement pour les environnements statiques. Nous souhaitons vérifier si les choix restent valides lorsque plusieurs agents sont instanciés dans un même environnement.

Avant de présenter ces tests, notons que nous avons dû réaliser un changement sur l'inertie. En environnement dynamique, il arrive qu'il n'y ait pas de solution valide pour un agent. En

cas de blocage, l'inertie ne doit pas intervenir dans le prochain cycle sinon elle encourage l'agent à s'immobiliser. Nous reviendrons sur ces problèmes liés à l'inertie dans les perspectives (chapitre V).

La figure IV.21 illustre un exemple de simulation comprenant vingt cinq agents. Nous avons représenté par un trait situé derrière chaque agent, les trajectoires suivies par eux lors des cycles précédents. Outre les évitements statiques, cette figure montre des exemples de croisements (passés et à venir) et de dépassements. Il faut noter que l'environnement utilisé pour les tests est encombré de nombreux obstacles et qu'en de nombreux endroits les agents n'ont pas la place pour se croiser. Il faut également noter qu'aucune hypothèse n'est faite sur les chemins suivis par les agents. Les agents ne possèdent pas une ordonnée curviligne privilégiée qui pourrait simplifier le problème.

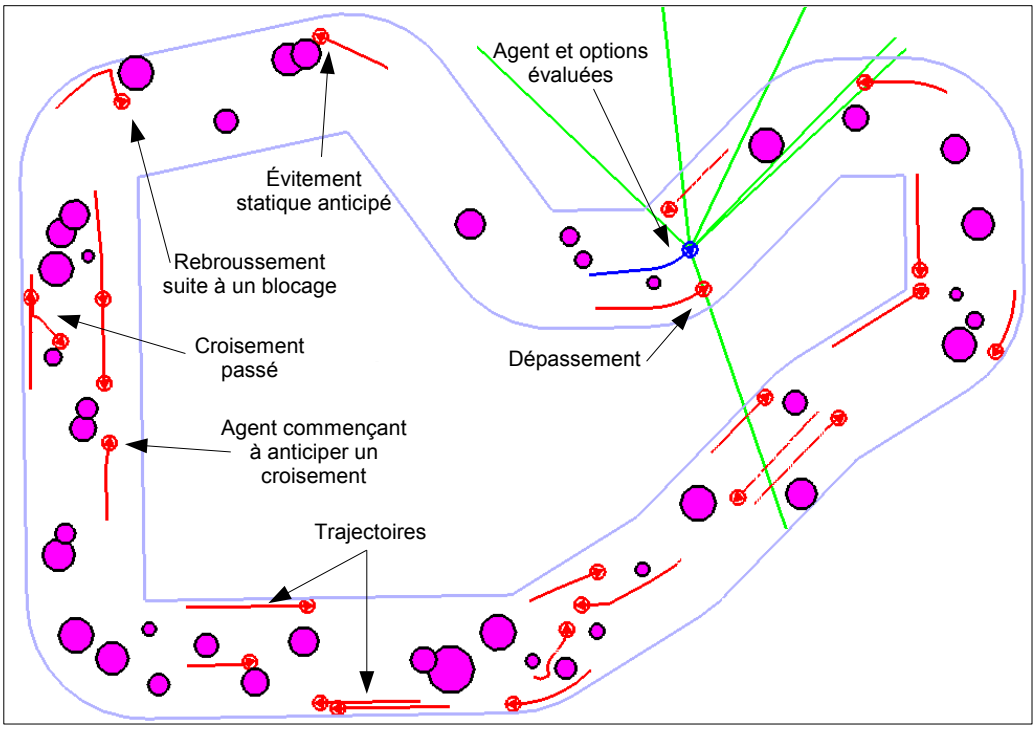

Figure IV.21 : *Exemple de simulation multi-agents.*

Lors des simulations, il apparaît que la suppression des pondérations a une influence importante sur le réalisme des interactions entre agents. Il est important que les agents anticipent suffisamment tôt leurs croisements sinon ils apparaissent trop réactifs. La figure IV.22 montre deux exemples. Sur la partie gauche de la figure, les agents ont anticipé leur croisement. Sur la partie droite, ils s'évitent au dernier moment. La suppression des pondérations peut donc s'avérer parfois problématique. Dans notre exemple, le vote d'un seul comportement d'évitement ne peut aller à l'encontre des votes des comportements « *inertie* » et « *suivre la voie* ». Lorsque les agents sont proches, ils s'évitent grâce au veto. Dans le chapitre V, nous présentons des perspectives (liées à l'inertie) utiles pour résoudre ce problème.

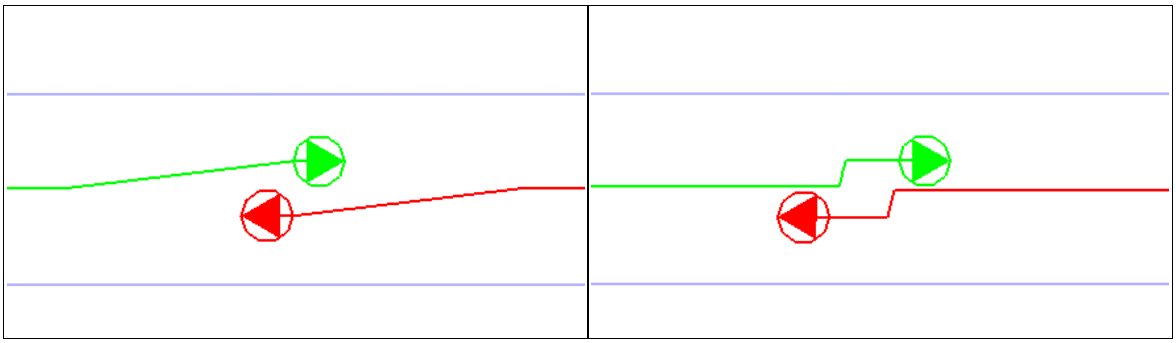

Figure IV.22 : *Anticipation ou non des croisements.*

En revanche, comme le montre la figure IV.23, un agent seul a tendance à s'écarter d'un groupe d'agents. Ce résultat est intéressant car selon [Thomas, 1999] il est conforme au comportement de piétons dans la réalité. De plus, il faut noter que ce résultat émerge sans que nous n'ayons programmé la notion de groupe et sa priorité face à un individu isolé. L'explication est que l'agent seul possède plusieurs comportements d'évitement correspondant à chaque agent du groupe. Ces comportements d'évitement votent pour dévier la trajectoire et emportent le vote. Pour les agents du groupe, l'unique comportement d'évitement d'agent ne parvient pas à s'opposer au vote des autres comportements (« *intertie* » et « *suivre la voie* ») c'est pourquoi les agents du groupes maintiennent leur trajectoire.

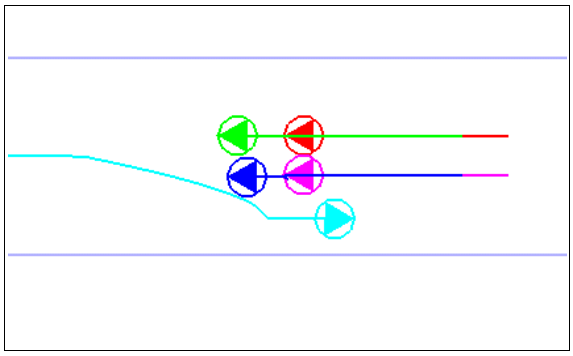

Figure IV.23 : *Agent seul s'écartant d'un groupe d'agents.*

Des blocages entre les agents peuvent apparaître durant les tests. Par exemple, sur la figure IV.24 un obstacle statique rétrécit la voie. Les agents ne se coordonnant pas, ils peuvent éventuellement se gêner mutuellement pendant plusieurs cycles. Comme nous allons le voir, les blocages se résorbent de moins en moins facilement avec l'augmentation de la densité d'agents.

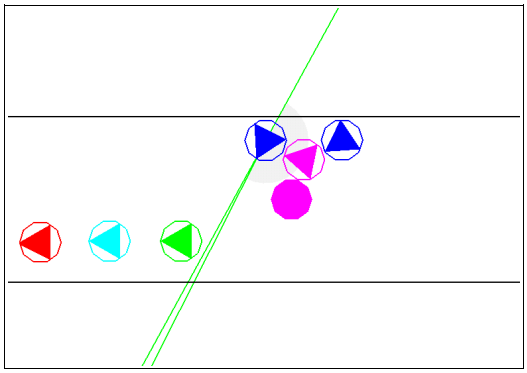

Figure IV.24 : *Blocage induit par un rétrécissement de la route.*

Lors des tests réalisés, nous avons augmenté progressivement la densité des agents. Les courbes présentées ci-dessous ont été obtenues en mesurant les performances d'un agent sur le circuit de la figure IV.21. Nous avons réalisé des simulations avec 10, 15, 20, 25, 30, 40 et 50 agents (positionnés aléatoirement). Pour un nombre d'agents donné, nous avons réalisé dix simulations de 10 000 cycles chacune (les courbes représentent des valeurs moyennes). Nous n'avons pas pu réaliser plus de 10 simulations à chaque fois car leur durée totale représente déjà environ 35 heures de calculs. Il faut environ 5 heures pour toutes les simulations de 10 à 25 agents et environ 30 heures pour toutes les simulations de 30 à 50 agents (sur un ordinateur de bureau de 2GHz). (Nous reviendrons sur ce point.)

La figure IV.25 montre l'évolution de la distance parcourue par un agent en fonction du nombre total d'agents. La distance totale parcourue diminue au delà de 20 agents. A partir de ce nombre, les modèles décisionnels trouvent difficilement de bonnes solutions. Les agents s'immobilisent plus souvent. La distance utile diminue plus rapidement, ce qui correspond au nombre croissant de manoeuvres réalisées par les agents pour éviter les obstacles.

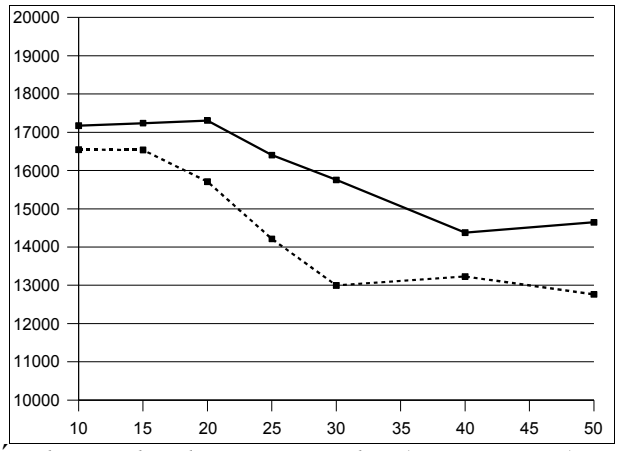

Figure IV.25 : *Évolution des distances totales (trait continu) et utiles (pointillés) en unité de distance, en fonction du nombre d'agents.*

La figure IV.26 illustre l'évolution du nombre de cycles pendant lesquels l'agent est considéré bloqué, en fonction du nombre total d'agents. Pour 50 agents, l'agent instrumenté passe le quart des cycles en situation de blocage. Il est assez difficile d'analyser cette courbe. En particulier, nous ne pouvons déterminer si ces blocages sont dûs aux limites du modèle ou à la complexité de l'environnement. Il faudrait disposer de données réelles pour voir comparer ces résultats au trafic dans une rue piétonne. Ce travail constitue une perspective. A ce stade de notre recherche, nous souhaitions évaluer les défauts du modèle dans les situations complexes. Dans le cadre de notre application, une vingtaine d'agents semble suffisante.

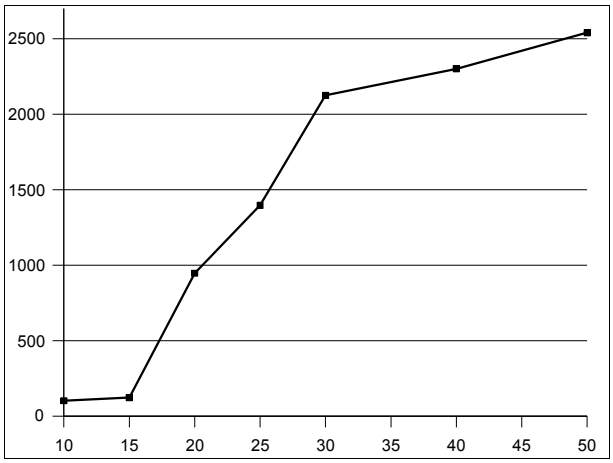

Figure IV.26 : *Évolution du nombre de blocages en fonction du nombre d'agents.*

La figure IV.27 illustre l'évolution de la variation moyenne de direction d'un agent, en fonction du nombre total d'agents. Celle-ci croit jusque 30 agents et décroît ensuite. En effet comme nous venons de le voir, pour un nombre important d'agents (entre 30 et 50), ceux-ci ont tendance à s'immobiliser. Ce qui réduit le nombre moyen de leurs hésitations.

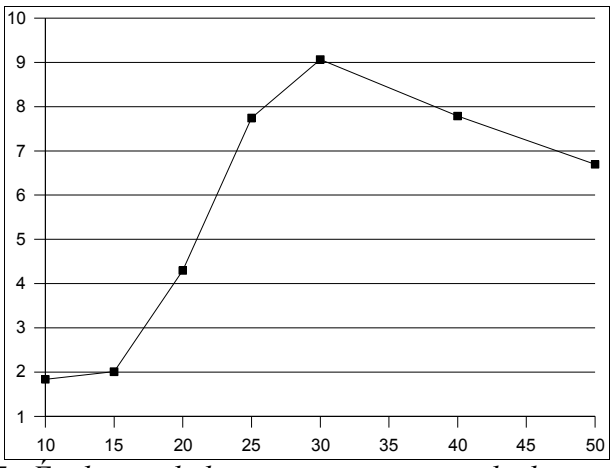

Figure IV.27 : *Évolution de la variation moyenne de direction (en degré) en fonction du nombre d'agents.*

La courbe IV.28 ci-dessous illustre l'évolution du nombre maximal (trait continu) et moyen (pointillés) d'options valides en fonction du nombre d'agents. La courbe montre que lors des simulations avec 50 agents, un agent peut évaluer jusqu'à 80 options lors d'un cycle (80 étant la moyenne sur 10 simulations). Dans le pire des cas, un agent a utilisé 37 comportements et proposé 113 options invalides et 119 options valides.

Le nombre important d'options et de comportements explique la durée excessive des simulations. L'implémentation naïve du modèle qui alloue dynamiquement les options renforce ce problème.

Afin d'atténuer ces défauts, nous envisageons d'adapter, en fonction de la situation, la distance au-dessous de laquelle les obstacles et autres agents sont pris en compte. En effet, lorsque la voie est encombrée, un piéton ne doit pas prendre en compte tous les piétons mais seulement ceux qui le gênent directement. En outre, il serait possible d'accélérer la perception des agents en utilisant une division de l'environnement (voir le paragraphe « perception des

acteurs virtuels » au chapitre I). Ces critiques ne remettent pas en cause le modèle décisionnel mais son application. Nous reviendrons sur ces critiques dans nos perspectives.

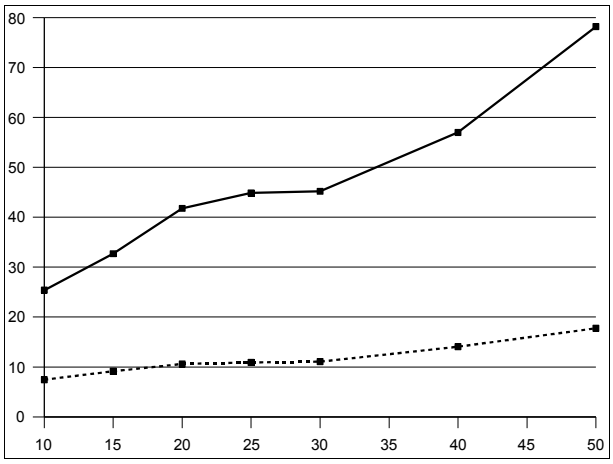

Figure IV.28 : *Évolution du nombre moyen (pointillés) et maximal (trait continu) d'options valides considérées par un agent en fonction du nombre total d'agents.*

Pour conclure la présentation des essais réalisés en multi-agents, il convient de souligner les points positifs qui n'ont pas été chiffrés. En particulier, nous n'avons pas reporté le nombre de collisions car il reste nul. De même, le nombre de sorties de route n'a pas été mentionné car elles sont extrêmement limitées et dues à un défaut mineur, identifié lors des tests en environnement statique. Les résultats sont discutés dans le paragraphe suivant.

3.6 Discussion des résultats

Afin d'analyser les résultats, il convient de distinguer les tests réalisés en environnement statique de ceux réalisés en environnement dynamique. En effet, dans notre démarche (et conformément au principe de récursion) nous avons considéré le problème de navigation selon trois niveaux d'analyse différents : comportement, agent (ou microscopique) et système multi-agents (ou macroscopique).

Les résultats en environnement statique sont importants car ils illustrent l'émergence de propriétés depuis le niveau comportement vers le niveau agent. Comme nous l'avons souligné dans la bibliographie, ce problème est déjà très complexe. Les résultats en environnement dynamique soulignent l'émergence de propriétés depuis les niveaux « comportement » et « agents » vers le SMA. En outre, la difficulté est que de la validation des comportements en environnement statique n'est pas suffisante pour obtenir de bons résultats dans un SMA (par exemple, il a été nécessaire de modifier le comportement « *inertie* »).

3.6.1. Résultats en environnement statique

Comme nous venons de le souligner, les tests en environnement statique permettent de valider les comportements ainsi que la procédure de vote. Nous avons comparé nos résultats à ceux de l'AFAG. Nous obtenons des résultats meilleurs pour la persistance des choix (variation moyenne de direction de 0,42° contre 0,64°) et des résultats comparables en ce qui concerne la réalisation des buts. Contrairement à l'AFAG, notre modèle respecte les contraintes inhérentes à la navigation (absence de collisions et de sorties de route). Il faut cependant rappeler que le respect des contraintes s'accompagne parfois de variations très brusques de direction (illustrées par la figure IV.20, page 119). Pour pallier ce défaut, il

conviendrait de modifier l'agent pour qu'il soit capable de tourner sur place en plusieurs cycles.

Outre les résultats que nous venons de rappeler, ces tests nous ont permis de souligner quelques particularités du mécanisme de vote.

Le premier point concerne l'influence des options. En effet, chaque option influe sur le vote. Par exemple, la présence d'options similaires divise les votes au profit d'autres options. Il n'est donc pas judicieux de proposer beaucoup d'options qui peuvent «polluer» le vote. Dans le cadre de la navigation, nous avons préféré ne pas proposer d'options relatives aux obstacles se trouvant derrière l'agent. A plus long terme, il conviendrait d'utiliser une méthode de vote qui limite l'influence des options perdantes.

Le second point concerne les compromis réalisés par le vote. Les tests réalisés montrent que si aucune précaution n'est prise, il est possible que le système sélectionne (comme pour la fusion) une action qui ne satisfait pas la majorité des comportements. En effet, dans certains cas les options qui ne satisfont aucun comportement peuvent être sélectionnées car elles ont été classées moyennes par tous les comportements (voir évitement d'obstacles). C'est pour cela qu'il s'avère parfois plus intéressant de classer les options en deux catégories (pré-ordre total) plutôt que d'établir un vrai classement (ordre total). Cela montre l'intérêt d'utiliser une méthode de vote qui prend en compte les pré-ordres. Il serait également intéressant d'essayer notre modèle avec une méthode utilisant un système de points avec bonus (cf. tableau III.1 page 68). En effet, les bonus peuvent défavoriser ces options qui, dans le cadre de la navigation, réalisent de mauvais compromis.

Le modèle décisionnel a ensuite été appliqué dans le cadre de simulations multi-agents.

3.6.2. Résultats en environnement dynamique

Les tests en environnement dynamique ont été réalisés dans des environnements encombrés d'obstacles. A de nombreux endroits, ces derniers ne permettent que le passage d'un seul agent à la fois. De plus, les agents simulés n'avancent pas tous dans le même sens, créant ainsi des conflits entre eux. Enfin, il convient de rappeler que nous n'avons fait aucune hypothèse permettant de simplifier la circulation des agents : ils ne possèdent pas d'ordonnée curviligne favorite ni de sens de circulation privilégié sur leur voie.

Les essais ont permis de mettre en évidence l'émergence d'un phénomène intéressant n'ayant pas été programmé (agent seul évitant un groupe). Mais ils ont également montré l'apparition de blocages dans les passages étroits. Ces blocages sont liés au fait que les agents ne se coordonnent pas. L'accroissement du nombre d'agents, lors des différentes simulations, a permis de mesurer les limites du modèle décisionnel. Les données mesurées montrent que les résultats sont corrects pour une vingtaine d'agents : ils réalisent leurs objectifs (avancer), les variations de choix (de direction) sont limitées, les temps de calculs sont raisonnables. Notons que disposer de 20 agents suffit pour notre application. En effet, le but n'est pas de simuler des foules mais de disposer de quelques personnages crédibles.

Outre les résultats que nous venons de rappeler, ces tests nous ont permis de souligner quelques défauts liés au nombre d'options, à l'utilisation du comportement « *inertie* », à l'abandon des pondérations.

Espace continu et nombre d'options

Sélectionner des options dans un espace continu est avantageux pour la pertinence des choix, la réduction des hésitations, etc. (cf. paragraphe 3.2). Afin de sélectionner les actions dans un espace continu, nous laissons les comportements proposer des options. Lors des tests en environnement multi-agents, nous avons mesuré qu'un agent propose et évalue jusqu'à 119 options valides. Ce nombre important d'options s'avère pénalisant car il augmente le temps de cycle du modèle décisionnel. Ce problème n'est pas très contraignant pour notre application (une vingtaine de piétons suffit). Nous avons donné quelques idées pour réduire le nombre d'options dans la cas de la navigation. Nous reviendrons sur ce problème lors de la présentation de nos perspectives (chapitre V) et proposerons des solutions plus générales.

Le comportement « *inertie* »

Le comportement « *inertie* » joue un rôle plus important que ce qui apparaît dans la bibliographie. Dans notre modèle, il ne limite pas seulement les hésitations mais permet de sortir l'agent de situations bloquantes. Il faut noter que dans notre modèle l'inertie prend une place prépondérante dans la sélection d'actions. Cette importance est variable car le nombre de comportements n'est pas fixe. Dans notre application, l'inertie représente au maximum la moitié des voix lorsqu'il n'y a pas d'obstacle statique, un tiers des voix si il y a un seul obstacle, etc.

Cette prépondérance de l'inertie peut avoir des inconvénients, notamment, lorsqu'elle s'oppose aux évitements d'obstacles qui sont sont alors plus anticipés. De même, si un comportement d'inertie trop simpliste est utilisé alors lorsqu'aucune solution n'est valide (et que l'agent reste sur place) alors l'inertie encouragera l'agent à ne plus bouger. Les tests réalisés nous ont donc montré qu'il n'est pas toujours judicieux de voter pour la solution retenue au pas précédent. Ils nous encouragent à reconsidérer l'utilisation de ce comportement. Nous revenons sur ce problème dans le chapitre V consacré aux perspectives

Pondérations

Les résultats présentés ont été obtenus sans utiliser de pondérations. Dans certains cas, il eût été plus simple d'y recourir. Par exemple, elles auraient permis de réduire l'importance de l'inertie (dans les cas où sa prépondérance s'oppose à l'anticipation d'un obstacle). Nous ne nions pas qu'elles possèdent une utilité, mais il ne nous semble pas souhaitable qu'elles prennent une place trop importante et injustifiée dans la sélection d'actions. Par ailleurs, comme nous l'avons vu, il est possible d'agir sur plusieurs paramètres avant de recourir à une pondération fixe et arbitraire.

4. Conclusion

Dans ce quatrième chapitre nous avons appliqué notre modèle décisionnel orienté comportement à la navigation autonome en environnement simulé. Nous avons présenté la navigation dans la première partie de ce chapitre. Cela nous a permis de lister les critères mesurables et les expérimentations à mettre en oeuvre pour montrer l'intérêt de notre modèle.

La seconde partie de ce chapitre est consacrée à l'implémentation du modèle et à la conception de comportements spécifiques. L'avantage de notre proposition est d'offrir une certaine souplesse pour la conception des comportements car il est possible d'agir sur les options, les veto et les évaluations. De plus, nous avons développé des comportements disposant d'une mémoire et pouvant échanger des informations (double veto). Le modèle décisionnel est donc légèrement plus complexe que les modèles semblables existants. Cela est dû à l'utilisation des veto, à l'abandon des pondérations et à la volonté de respecter les contraintes (collisions, etc).

La troisième partie est consacrée aux évaluations. La difficulté de la validation et de l'amélioration du modèle réside dans l'existence de trois niveaux d'analyse : le système multi-agents, l'agent et le comportement.

Les tests en environnement statique nous ont permis de comparer notre modèle à un modèle discret ainsi qu'à l'AFAG. Nous obtenons de meilleurs résultats pour la persistance des choix et des résultats comparables en ce qui concerne la réalisation des objectifs. En outre, l'avantage principal de notre modèle est de respecter les contraintes.

Les tests en environnement multi-agents montrent l'apparition d'un phénomène émergent (évitement d'un groupe d'agents). Ils montrent également les limites du modèle de navigation. Elles se traduisent, en particulier, par l'apparition de blocages dûs aux comportements d'évitements inter-agents trop réactifs. Bien qu'il semble impossible d'éviter l'apparition de blocages, il convient d'essayer de les résorber rapidement. Les simulations soulignent des problèmes liés à l'anticipation des évitements et nous encouragent à reconsidérer l'utilisation du comportement « *inertie* ».

Nous proposons dans le chapitre V de prendre en compte ces remarques et d'esquisser quelques pistes d'amélioration du modèle de navigation et du modèle décisionnel général.

Chapitre V : Perspectives de recherches

Dans le chapitre précédent nous avons montré les limites du modèle de navigation et du mécanisme de sélection d'actions. La navigation dans un environnement dynamique et continu et la sélection d'actions sont des problèmes complexes, traités depuis longtemps. Les perspectives sont donc très nombreuses, pour l'application présentée et pour le modèle lui même. Ce chapitre comporte deux principales sections présentant les perspectives relatives à ces deux aspects.

La première partie de ce chapitre présente les perspectives liées à l'application. Nous aborderons celles liées au développement de piétons virtuels dans le cadre d'environnements plus réalistes que ceux utilisés pour la validation du modèle. Ensuite, nous reviendrons sur les interactions entre agents. Nous aborderons les problèmes liés aux croisements de flux de personnages et proposerons quelques pistes destinées à prévenir les blocages ou à les résorber.

De même, nous n'avons pas traité la simulation de groupes de piétons. Nous estimons que cette application est intéressante car elle permettra de rendre les simulations plus réalistes. Par ailleurs, les interactions mises en jeu lors de la navigation en groupe sont assez complexes et donc intéressantes d'un point de vue théorique pour les SMA.

La seconde partie du chapitre est consacrée aux perspectives de recherches générales et indépendantes de toute application. Nous reviendrons sur le comportement « *Inertie* » pouvant être utilisé dans divers domaines. Il conviendra de montrer qu'il peut être amélioré afin de permettre une remise en cause du choix, lorsque cela s'impose. Le nombre d'options proposées par le modèle s'avère parfois trop élevé. Nous proposons une idée qui limiterait le nombre d'options lorsqu'il devient trop pénalisant. Nous revenons également sur la notion de veto, très utile mais pouvant être problématique, notamment, lorsque les veto invalident toutes les options proposées. La dernière perspective n'est pas la moins importante ni la moins ambitieuse et concerne l'adjonction de capacités cognitives utiles lors des défaillances du modèle actuel.

Nous terminerons ce chapitre par la présentation des autres domaines auxquels nous souhaiterions appliquer notre modèle décisionnel.

1. Perspectives liées à la simulation de piétons dans l'environnement RESPECT

Dans cette section, nous traiterons des perspectives relatives à l'utilisation du modèle de navigation pour des piétons virtuels. Elles concernent d'un part, les possibilités et les difficultés nouvelles liées au développement de piétons dans des simulations plus proches de la réalité. D'autre part, nous aborderons des perspectives liées aux interactions entre les agents.

1.1 Simulation de piétons dans des environnements plus réalistes

Afin de réaliser les premiers essais de notre modèle, nous avons utilisé un environnement simplifié. Nous avons obtenu des résultats encourageants, il est important d'élargir notre cadre d'étude. Grâce au projet RESPECT, nous disposons maintenant d'un environnement urbain simulé paramétrable (description de la ville), graphiquement agréable, simulant des véhicules, etc. La simulation dans un environnement plus réaliste offre de nouvelles possibilités et de nouveaux défis.

Les piétons simulés dans l'environnement RESPECT se déplacent dans des villes. Ils disposent donc de fonctions leur permettant de planifier un itinéraire. Cela est rendu possible par une modélisation appropriée de la ville. Il conviendra donc de lier notre module de navigation réactive avec un module de planification. Cette perspective applicative est à rapprocher d'une perspective plus générale destinée à introduire des capacités cognitives dans le modèle décisionnel. Nous reviendrons donc sur ce point.

Dans une application plus complète les piétons se déplacent sur les trottoirs et traversent la chaussée sur laquelle se déplacent des véhicules. Nous ajouterons les comportements correspondants pour que nos piétons soient capables de traverser la chaussée, dans les cas simples (passage protégé avec feux) ou plus complexes (traversée hors des passages protégés).

Dans la réalité, les piétons tendent à circuler sur le bord droit du trottoir. En effet, les piétons faisant face au trafic acceptent d'en être proches, alors que les autres s'en éloignent. (Pour plus de détails sur les habitudes des piétons voir [Granié et al., 2001].) Pour prendre en compte cette remarque, nous pouvons modifier le comportement des piétons en ajoutant une ordonnée curviligne préférentielle. Notons que cela devrait simplifier les évitements entre les piétons. De plus, dans les essais que nous avons réalisés, nous n'avons pas autorisé les piétons à quitter leur trottoir. Dans la réalité, ils le quittent lorsqu'il est encombré d'obstacles statiques ou quand les piétons sont trop nombreux. Pour notre modèle, il conviendrait donc d'ajouter des méthodes permettant de relâcher les contraintes sur le comportement « *suivre une voie* » lorsqu'il est possible de descendre sur la route sans risque.

Ces possibilités supplémentaires nous permettront également de créer des piétons aux comportements différenciés. Cela se traduira par la modification de paramètres relatifs aux nouvelles possibilités. Par exemple, pour la traversée, un piéton peut utiliser systématiquement les passages protégés ou traverser librement la chaussée.

L'utilisation de notre modèle dans un environnement plus réaliste nous amènera également à faire cohabiter nos piétons virtuels avec les avatars de joueurs. Nous pensons que l'introduction de joueurs ne devrait pas remettre en cause le modèle décisionnel des piétons car ces derniers ne font pas d'hypothèse sur les réactions des autres piétons. Il convient de

réaliser des simulations pour le vérifier. De même, nous aimerions tester notre modèle de navigation dans des systèmes multi-agents hétérogènes. Notamment, nous souhaiterions réaliser des tests avec des agents moins réactifs comme ceux présentés dans [Airault et al., 2004].

Lors de nos simulations, nous n'avons pas complètement traité tous les problèmes et possibilités qui concernent les interactions entre agents. Nous reviendrons donc sur les évitements et introduirons une nouvelle interaction : le déplacement en groupe.

1.2 Amélioration des interactions entre les agents

Les tests réalisés montrent que des blocages apparaissent dès que deux agents (ou plus) entrent en conflit spatial. Un conflit spatial est une situation où plusieurs agents tentent d'utiliser simultanément un même espace alors que celui-ci est insuffisant pour le permettre [Simonin, 2001]. Le nombre de conflits spatiaux augmente avec la densité des obstacles et des agents, dégradant leurs performances et bloquant le système. Nous n'avons pas explicitement mis en oeuvre de méthode pour prévenir ces blocages. De plus, nous n'avons rien proposé pour les résoudre : ceux-ci se résorbent d'eux-mêmes après un certain temps mais il serait intéressant d'accélérer leur résolution.

De plus, les nouvelles possibilités (traversées) offertes par un environnement évolué peuvent apporter de nouvelles causes de conflits spatiaux. C'est pour cela qu'il conviendra de porter une attention particulière sur les interactions entre agents. Nous étudierons les croisements de flux de personnages. Enfin, pour être plus réalistes, les simulations devront comporter des groupes de quelques piétons se déplaçant ensemble.

1.2.1. Croisement de flux de personnages

Dans ce rapport, nous avons concentré nos tests sur la navigation d'agents se déplaçant sur une même voie. Nous n'avons pas considéré le cas des croisements de flux d'agents. Or, ce cas n'est pas rare. L'intersection de rues piétonnes en est un exemple.

Notre modèle permet de prendre en compte ce problème. La figure V.1, ci-dessous est une capture des essais réalisés pour l'article [Hanon et al., 2006] et présente des croisements de flux d'agents. Ceux-ci prennent en compte la direction de l'agent évité. Ils favorisent la collaboration en privilégiant, lorsque cela est possible, les trajectoires qui ne coupent pas la route de l'agent évité.

Figure V.1 : *Trajectoires d'agents à un croisement.*

Dans le futur, nous souhaitons étendre notre modèle afin de prendre en compte les possibilités inhérentes à ces situations de croisements ou de jonctions de flux d'agents. Par exemple, il conviendra que les comportements proposent et évaluent de nouvelles options permettant soit de laisser passer les agents évités, soit d'accélérer pour passer devant eux.

Même s'ils disposent de possibilités d'interactions plus évoluées, ces situations introduiront nécessairement de nouvelles sources de conflits spatiaux et de blocages.

1.2.2. Prévention et résolution *a posteriori* des blocages

Prévention des blocages

L'apparition de blocages s'avère extrêmement pénalisante, d'autant que l'arrivée de nouveaux agents a tendance à accroître les blocages. Dans le cadre de simulations participatives, il n'est pas acceptable d'arrêter la simulation parce que la congestion du trafic empêche les participants de continuer.

Dans [Doniec et al., 2006] l'auteur ajoute à son modèle de navigation un module permettant à ses agents de détecter les éventuels conflits et de les prévenir. Dans son application de simulation de trafic, cela permet aux voitures d'éviter de s'engager dans un carrefour congestionné. Adapté à notre application, ces travaux permettraient aux agents de ne pas s'engager dans des zones où il existe déjà des conflits entre d'autres agents.

Bien que les agents doivent chercher à les éviter, il faut également qu'ils soient capables de résoudre ces problèmes de blocage.

Résolution des blocages

Bien qu'il soit souhaitable de les éviter *a priori,* il est inévitable que des conflits spatiaux apparaissent dès que la densité des agents est localement importante. Dans ce cas, il s'agit de mettre en oeuvre une méthode pour les résorber rapidement. Nous envisageons de nous inspirer du modèle « satisfaction-altruisme » [Simonin, 2001] qui permet de traiter ces problèmes simplement, même si le nombre d'agents est important. Nous rappellerons donc

brièvement le principe de ce modèle en montrant qu'il peut être appliqué à notre modèle décisionnel.

Dans [Simonin, 2001], lorsque des agents sont en conflit, ils communiquent[28] leur insatisfaction par l'intermédiaire de signaux locaux simples. Les agents calculent leur insatisfaction en fonction de leur paralysie et leur niveau de contrainte. Ces deux données peuvent être calculées simplement. L'évaluation de la paralysie est triviale pour l'application. Dans le cadre de notre modèle, le niveau de contrainte peut être fonction des veto et du nombre d'options valides. Simonin part du principe que pour débloquer une situation, les agents les moins contraints doivent adopter le comportement le plus altruiste. Si un agent reçoit un signal d'insatisfaction plus fort que le sien alors il devient altruiste. Un agent altruiste oublie ses buts personnels et accepte de s'écarter de l'agent qu'il gêne. Pour que l'agent altruiste s'écarte, une force de répulsion est utilisée (cf. chapitre II). Nous n'entrerons pas dans les détails du modèle qui prévoit, en plus, les mécanismes nécessaires à la persistance de l'altruisme et à sa propagation aux autres agents.

Dans le cadre d'une sélection par le vote, la modélisation d'une force de répulsion reste simple. Le comportement d'évitement peut proposer une option de vitesse quelconque se dirigeant dans la direction opposée à l'agent évité. Le vote de ce comportement peut également être modifié afin de favoriser les options qui maximisent la distance entre les deux agents. Le vote peut s'exprimer avec la formule V.1 suivante.

$$AA'_{t+1}(o_i) > AA'_{t+1}(o_j) \Rightarrow o_i > o_j$$

avec : $o_i > o_j$ signifiant que l'option o_i est préférée à o_j

$AA'(o_k)$ la distance entre l'agent et l'agent évité, en fontion de l'option k

$$AA'(o_k) = \sqrt{(x_o - x_a + vo_{k,1}\cos(vo_{k,0}))^2 + (y_o - y_a + vo_{k,1}\sin(vo_{k,0}))^2}$$

(formule V.1)

Dans le cadre de notre modèle, l'utilisation du modèle « satisfaction-altruisme » semble réalisable. La principale difficulté est d'intégrer la réaction altruiste dans le modèle afin que l'agent ne tienne plus compte de ses objectifs personnels. Il conviendra alors de changer le déroulement du vote pour ne plus prendre en compte les préférences de certains comportements. Il sera également nécessaire de vérifier qu'il est possible de se passer des pondérations (utilisées, par le modèle de Simonin, lors de la fusion). Enfin, il conviendra de montrer les avantages et inconvénients d'une sélection par le vote dans le cadre de l'utilisation du modèle « satisfaction-altruisme ».

Nous avons présenté quelques pistes pour résoudre les problèmes de conflits spatiaux intervenant entre des agents isolés. Il convient d'aborder également le problème du déplacement en groupe.

28 L'introduction de communications simples entre les agents ne constitue pas une hypothèse irréaliste car il existe vraisemblablement une forme de communication entre les piétons « réels » en interaction.

1.2.3. Simulation de groupes de piétons

Il est important d'aborder la simulation de groupes de deux ou trois piétons comme cela se produit dans la réalité. Par ailleurs, ce problème complexe est intéressant d'un point de vue théorique pour les systèmes multi-agents. En effet, il conviendra d'aborder les interactions entre les agents d'un même groupe, ou de groupes différents ainsi que les interactions entre les groupes et les agents isolés (cas que nous avons déjà commencé à aborder).

Dans ce cadre, le problème de sélection d'actions est plus complexe. En effet, dans un groupe, les individus adaptent leur comportement pour prendre en compte leurs contraintes personnelles et également celles des autres. Par exemple, lors de l'évitement d'un obstacle, un agent devra dévier sa trajectoire pour l'éviter tout en prenant soin de ne pas gêner les autres membres de son groupe.

Le problème de simulation de groupes de piétons a déjà été abordé par Terry Hostetler [Hostetler et al., 2002]. Dans le domaine connexe de la robotique, les problèmes similaires de navigation de robots en formation ont déjà été traités à maintes reprises (voir par exemple [Arkin, 1998] ou [Arnaud, 2000]).

La figure V.2 tirée de [Hostetler, 2003] montre l'exemple d'un groupe d'agents rapides doublant un groupe plus lent tout en évitant un obstacle statique. Comme le montre la figure, les agents d'un même groupe suivent un même objectif. Les agents ne sont pas totalement responsables de leurs choix puisqu'il existe une forme de décision centralisée sélectionnant l'objectif commun. De même, dans les applications de robotique, des rôles de leader et de suiveurs sont attribués aux robots afin de simplifier le problème de cohésion de la formation.

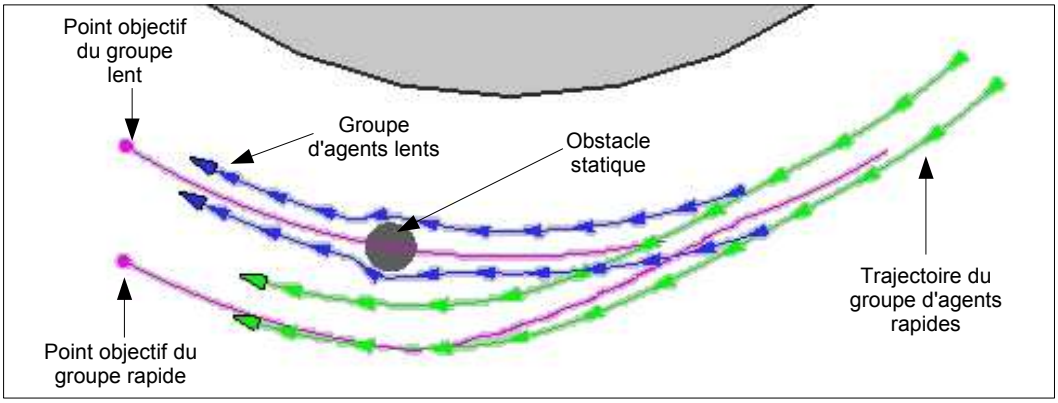

Figure V.2 : *Groupe d'agents rapides doublant un groupe plus lent [Hostetler, 2003].*

Le propre des simulations multi-agents est de distribuer les décisions. Par conséquent, même si l'utilisation de décisions centralisées au niveau d'un groupe donne de bons résultats, elle n'est pas pleinement satisfaisante. Il serait donc intéressant de vérifier s'il est possible de rendre chaque agent du groupe responsable de sa décision.

Ce problème de navigation en groupe constitue donc un problème intéressant, à la fois dans le cadre applicatif mais aussi d'un point de vue plus théorique pour les systèmes multi-agents. Nous consacrons la fin de ce chapitre aux perspectives liées au modèle décisionnel générique pouvant être appliqué à divers domaines.

2. Perspectives liées au modèle de sélection d'actions

Contrairement à ce que nous avons envisagé précédemment, les perspectives que nous présentons dans cette partie sont indépendantes de l'application et concernent l'architecture proposée.

La première perspective concerne le comportement « *inertie* ». La seconde est relative à la maîtrise du nombre d'options étudiées par chaque comportement lors du vote. La troisième concerne les veto. La quatrième perspective, que nous envisageons sur un plus long terme, traite de l'adjonction de capacités de raisonnement symbolique dans le modèle décisionnel. Enfin, le modèle que nous proposons étant générique, nous terminerons nos perspectives par la présentation des domaines auxquels nous souhaiterions l'appliquer.

2.1 Le comportement « *inertie* »

Le comportement « *inertie* » se situe à la limite entre le modèle générique et son application. En effet, c'est un comportement particulier, utilisé par l'architecture, mais pouvant être utile dans de nombreuses applications. Comme nous l'avons montré dans le chapitre IV, l'utilisation de ce comportement peut poser quelques problèmes.

Dans notre modèle, le comportement « *inertie* » joue un rôle plus important que ce qui apparaît dans la bibliographie. En effet, il ne limite pas seulement les hésitations mais permet de sortir l'agent de situations bloquantes. Il faut noter que pour notre modèle, l'inertie possède un poids prépondérant dans la sélection d'actions. Cette prépondérance est, cependant, variable car le nombre de comportements n'est pas fixe. Par exemple, pour la navigation, le vote de l'inertie représente la moitié des voix lorsqu'il n'y a pas d'obstacle statique, le tiers des voix si il y a un seul obstacle, etc.

Cette place centrale de l'inertie peut avoir des inconvénients. En particulier, car elle s'oppose parfois à des changements de décisions s'avérant nécessaires. Le principal défaut que l'on puisse reprocher à l'inertie est de ne pas tenir compte de l'évolution de l'environnement. En effet, nous nous plaçons dans l'hypothèse d'un environnement dynamique et bien qu'il soit important de persévérer dans des choix, il est également important de savoir quand les remettre en cause.

Lors de changements notables de l'environnement, il nous semble évident de devoir accorder moins d'importance au choix réalisé précédemment car il est relatif à un contexte différent. Dans le cadre de notre modèle, les changements importants de l'environnement sont facilement détectables. En effet, ils correspondent à l'instanciation ou à la destruction de comportements du modèle décisionnel. Lorsqu'un de ces changements de l'environnement a été détecté, il est possible, soit de ne pas tenir compte du vote de l'inertie, soit d'affecter à son vote un coefficient de pondération inférieur à un.

Dans le cadre de la navigation, cette idée permettrait une anticipation plus systématique des évitements d'obstacle. Cette idée constitue le début d'une réflexion mais il convient de l'appliquer et de l'évaluer pour vérifier qu'elle n'introduit pas de défauts dans le modèle décisionnel.

Le paragraphe suivant traite d'un perspective destinée à limiter le nombre d'options évaluées par le modèle et réduire ainsi son temps de cycle.

2.2 Maîtrise du nombre d'options évaluées

Les tests réalisés au chapitre IV ont montré que le nombre d'options proposées par les comportements peut être relativement important (jusqu'à environ 120 options pour une simulation avec 50 agents). Nous avons donné quelques idées destinées à réduire ce problème dans notre application, en limitant le nombre d'agents pris en compte par le modèle, par exemple.

Quelle que soit l'application considérée, les temps de calculs peuvent devenir prohibitifs lorsque que le nombre d'options est trop important. Par ailleurs, les résultats des votes peuvent être affectés. Par exemple, des options semblables peuvent diviser les votes au profit d'autres options.

Afin de réduire le nombre d'options, nous proposons de regrouper les options proches. En effet, les options étant définies par des vecteurs de réels, il est possible de calculer la distance qui sépare deux options. Ces distances pourraient être utilisées afin de déterminer des groupes d'options proches et réduire ainsi l'ensemble des options candidates au vote.

Outre le problème de réalisation de ces groupes, l'une des questions inhérentes à cette proposion est de savoir à quelles conditions il est utile « d'investir » le temps nécessaire à la réduction du nombre d'options pour raccourcir le temps de cycle de l'agent.

2.3 Réflexion sur la place des veto

Les veto constituent un point important du modèle décisionnel car ils assurent le respect des contraintes. Nous proposons deux perspectives liées à ces veto. La première concerne leur application et la seconde le cas dans lequel aucune option ne respecte les veto.

2.3.1. Invalidation des options par les veto

Les veto invalident certaines options. Nous nous demandons s'il serait intéressant de modifier les options à la place de les supprimer. L'inconvénient majeur de cette proposition est d'augmenter le nombre d'options car, pour une option invalide, il faudra proposer plusieurs éventualités se situant aux limites du veto. Nous n'avons pas envisagé cette possibilité car dans notre application, les comportements qui déposent des veto proposent également les options qui permettent de les respecter. Par exemple, le comportement d'évitement propose les options qui contournent l'obstacle source du veto. Cette possibilité peut être utile pour certaines applications. En particulier pour les cas où les veto invalident toutes les options.

2.3.2. Invalidation de l'ensemble des options

Après avoir appliqué les veto à l'ensemble des options, il est probable qu'aucune des options ne soit valide. [Sukthankar, 1997] met en oeuvre un algorithme qui tente de prévenir cette situation mais qui peut se produire malgré tout. On peut distinguer deux éventualités : soit aucune option acceptable n'a été proposée, soit les veto bloquent totalement l'espace d'actions, c'est à dire que le problème est sur-contraint. Il est possible d'effectuer un test afin de déterminer quelle est l'éventualité courante.

Dans le cas où aucune option valide n'a été proposée, il faut être capable de trouver de nouvelles options qui satisfassent les contraintes exprimées par les veto.

Le cas où le problème est sur-contraint est plus complexe mais est susceptible de se produire plus souvent. Il est possible d'envisager de ne pas tenir compte de certains veto,

autrement dit de relâcher des contraintes. Par exemple, pour la navigation, lors de l'évitement inter-agents la vitesse maximale des agents est prise en compte. Il doit donc être possible de relâcher la contrainte pour aller frôler l'agent à éviter et ainsi continuer à avancer malgré les veto.

Cette perspective, liée au veto et utilisant des notions proches des problèmes de satisfaction de contraintes, nous amène vers une perspective plus générale : l'adjonction de capacités cognitives aux sein de notre architecture.

2.4 Adjonction de capacités cognitives

Jusqu'à présent nous avons suivi une approche résolument orientée comportement. Cependant, il reste difficile de résoudre tous les problèmes de sélection d'actions auxquels un agent est confronté avec une architecture comportementale. Par ailleurs, utiliser une approche classique s'avère plus simple et plus efficace pour certains problèmes. La nécessité de construire des architectures décisionnelles qui intègrent des modules de différents niveaux d'abstraction s'est imposée (voir, entre autres, [Brooks et al., 1998]). Cependant, cet axe de recherche est difficile. Nous l'envisageons donc comme une perspective à plus long terme.

Dans le cadre du projet RESPECT [Aubert et al., 2003], un algorithme du plus court chemin permet de construire un itinéraire dans la ville (modélisée par un graphe). Celui-ci est suivi par le module réactif, en lui fixant comme objectif le point clef suivant dans l'itinéraire. Ainsi la description topologique de la ville est utilisée pour construire un itinéraire, adapté par le module réactif pour prendre en compte l'aspect dynamique de l'environnement.

L'exemple que nous avons utilisé pour introduire cette perspective est assez simpliste mais significatif de la complémentarité des approches. Les problèmes de communication et de transfert de contrôle entre les modules restent les principales difficultés à surmonter. Nous présenterons rapidement les modèles « *TouringMachines* » [Fergusson, 1992] et « *InteRRaP* » (Integration of Reactive behavior and Rational Planing) [Müller, 1996] (cf. figure V.3) car ils sont significatifs des principales méthodes existantes[29].

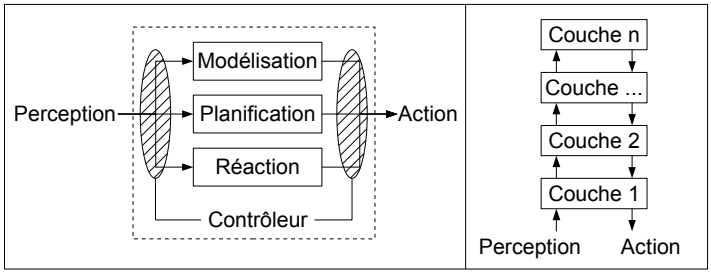

Figure V.3 : *TouringMachines [Fergusson, 1992] (à gauche) et InteRRaP [Müller, 1996] (à droite).*

L'architecture « *TouringMachines* » [Fergusson, 1992] (partie gauche de la figure V.3) est divisée en trois modules : Modélisation, Planification et Réaction. La sélection est similaire aux modèles orientés comportement car chaque couche suggère une action. Le problème de sélection d'actions est donc traité par le contrôleur. Celui-ci utilise des règles (si-alors) qui modifient les perceptions et les suggestions d'actions. L'architecture *TouringMachines* s'apparente donc aux architectures orientées comportement qui mettent en oeuvre des comportements complexes de planification (comme pour [Rosenblatt, 1996]). De même, le

29 Il existe de multiples architectures de ce type. Il convient cependant de citer le modèle « Skill Rule Knowledgede » [Chaib-draa, 1996] fondé sur le modèle de [Rasmussen, 1983].

contrôleur s'apparente au mécanisme de coordination de comportements. Fergusson réalise d'ailleurs un parallèle entre son travail et l'architecture de subsomption [Brooks, 1986]. « *InteRRaP* » [Müller, 1996] (Partie droite de la figure V.3) est divisée en de multiples couches dont les plus hautes sont les plus abstraites. Une couche active la couche supérieure (plus abstraite) si elle n'est pas capable de traiter le problème. En retour, une couche ordonne à une couche inférieure d'exécuter des actions.

L'architecture *TouringMachines* est très proche des architectures orientées comportement. Il ne nous semble pas approprié de s'inspirer de cette architecture pour améliorer notre modèle car nous aboutirons aux mêmes problèmes. Une architecture verticale semblable à *InteRRaP* nous semble donc plus adaptée dans ce cas précis. Dans le cadre de notre modèle, l'architecture globale à développer se présenterait alors comme sur la figure V.4 : un système à base de comportements complété d'un couche cognitive. Le contrôle final de l'agent serait donc laissé aux comportements.

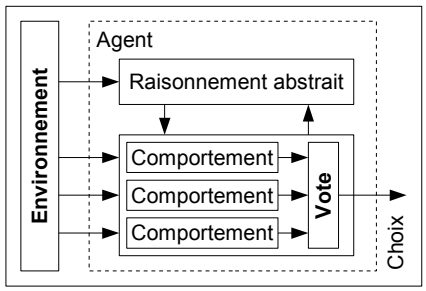

Figure V.4 : *Architecture globale composée de comportements et d'un module de raisonnement abstrait.*

Le rôle du module de raisonnement abstrait serait multiple. Comme nous l'avons vu dans l'exemple, il pourrait être capable de planifier des actions et agirait sur les comportements en leur fixant des objectifs ou en inhibant certains. Comme dans *InteRRaP,* le module de raisonnement abstrait pourrait être activé par les comportements lorsqu'ils détectent une anomalie (une égalité entre les options, pas d'option valide, trop d'options à évaluer, etc.). Il serait également intéressant, comme dans [Grislin et al., 2005] que la couche abstraite détecte elle-même les inconsistances dans les comportements. Les échanges entre couches doivent discutés car c'est sur ce point que se situe l'un des principaux « verrous technologiques ». Nous aborderons les possibilités de communication entre le module abstrait et les comportements.

L'échange d'informations du plus abstrait vers le comportemental est souvent effectif. Il peut, par exemple, permettre de fixer les objectifs des comportements qui gèrent le court terme. Il peut également permettre d'adapter la sélection d'actions : en modifiant les poids des comportements [Rosenblatt, 1996] ou en modifiant l'espace d'actions [Hostetler, 2003]. [Scheutz, 2004] propose de modifier la méthode de coordination (arbitrage, fusion) en fonction du contexte. Cette proposition part du constat qu'un arbitrage est parfois plus adapté. L'idée d'adapter la méthode de coordination est intéressante mais elle reste complexe et l'article ne propose pas de stratégie concrète pour définir quand elle doit être modifiée. Il n'existe d'ailleurs pas d'étude systématique montrant quel type de sélection est adaptée à un problème concret. D'autres pistes pourraient également être explorées, la modification de l'état ou de la réponse de comportements. La figure V.5 présente une illustration des influences possibles, du module abstrait sur les comportements et sur le système de coordination.

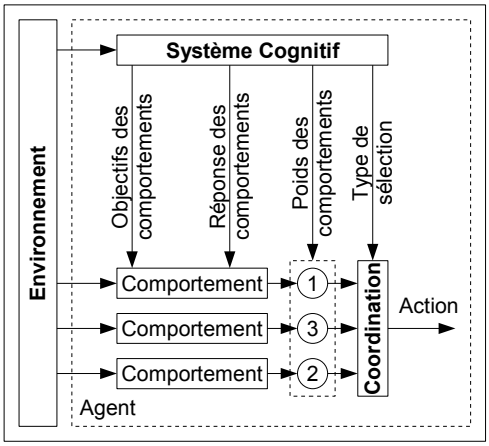

Figure V.5 : *Exemple de communication du module abstrait vers les comportements.*

La remontée des informations des comportements vers la couche plus abstraite est moins couramment développée. Dans *InteRRap* [Müller, 1996] une couche doit détecter qu'elle est incompétente en cas de problème et faire appel à la couche plus abstraite. Cela nous semble difficile à mettre en oeuvre dans le cadre d'un modèle tel que présenté en figure V.4 car il existe une distribution entre les comportements. C'est pourquoi nous pensons que la couche abstraite devrait détecter elle-même les inconsistances dans les comportements [Grislin et al., 2005]. De par sa position en dehors du système comportemental, la couche abstraite est capable de détecter les comportements ayant des objectifs contradictoires. De même, elle est capable de détecter les hésitations résultant de la sélection d'un comportement puis d'un autre, etc. Les dénominations que nous avons ajoutées aux options devraient également permettre au module abstrait de raisonner sur les options (qui sont des données numériques). Enfin, l'un des enjeux est d'exploiter ces informations pour aider la couche composée de comportements qui contrôle l'agent *in fine*.

Nous estimons aussi que les communications entre comportements constituent un point intéressant. Cette possibilité a été exploitée dans le cadre de la navigation mais devrait être élargie dans le cadre d'une architecture plus évoluée. En effet, nous avons introduit une communication symbolique indirecte entre les comportements. Elle est fondée sur l'analyse des veto et la détection de « doubles veto » (cf. chapitre IV). Plus généralement, les veto déposés peuvent être utilisés par les comportements. Ils pourraient par exemple, détecter que la solution qu'ils essaient de faire adopter à l'agent n'est pas bonne et qu'ils devraient d'eux-mêmes opter pour une autre possibilité.

Les possibilité d'extensions sont donc nombreuses. Notre dernière perspective concerne l'application de notre modèle à d'autres domaines.

2.5 Application à d'autres domaines.

Nous avons voulu développer un modèle générique qui puisse être utilisé dans d'autres contextes. Bien que les acteurs virtuels et la navigation en environnement virtuel soit une application intéressante, nous envisageons divers domaines d'applications comme la robotique mobile ou les animats.

La robotique mobile pourrait être un bon domaine d'application du modèle. Cette perspective semble légitime car le vote de préférences distribuées [Rosenblatt, 1996] [Sukthankar, 1997] et les architectures orientées comportement en général [Brooks, 1986] proviennent de la robotique. Malgré cela, cette perspective est assez ambitieuse au vu des

problèmes liés à l'identification des objets et à leur reconnaissance d'un cycle sur l'autre. L'étape de perception « grossière » qui permet d'instancier les comportements ne serait plus triviale. Il faudrait qu'elle reste relativement rapide (et donc imprécise) pour ne pas revenir aux problèmes inhérents aux architectures classiques fonctionnelles. De même, la perception ne pourrait plus se contenter, comme dans la simulation, de retourner une liste de pointeurs.

Le domaine des animats pourrait également être un cadre intéressant pour notre modèle. Il faut d'ailleurs noter que l'architecture de [Rosenblatt et al., 1989] a été adaptée dans ce contexte par [Tyrrell, 1993]. L'intérêt premier de ce cadre est de pouvoir fournir plusieurs objectifs au modèle décisionnel (boire, manger, chasser, dormir, etc.). De plus, dans ce cadre, il est possible de définir un plus grand nombre de relations entre les agents : collaboration ou, au contraire, compétition (proies et prédateurs).

Enfin, nous pensons qu'il serait intéressant d'aborder la problématique des simulations de foules. Nous sommes, cependant, conscients qu'il faudrait, pour cela, apporter de nombreuses modifications au modèle de navigation existant.

3. Conclusion

Dans ce chapitre, nous avons présenté les perspectives que nous envisageons à ce stade de notre recherche. Elles prévoient des améliorations au niveau de simulation de piétons virtuels (utilisation dans des environnements réalistes, interactions entre les piétons, simulations de groupes). Bien que concernant une application précise, ces perspectives présentent des cadres intéressants pour des recherches théoriques sur les systèmes multi-agents (interactions entre groupes d'agents, systèmes hétérogènes, etc.).

Nous prévoyons également d'améliorer notre modèle (comportement « *inertie* », réduction du nombre d'options, place des veto). Nous envisageons également d'étendre notre modèle avec un module de raisonnement symbolique. Ce module pourrait venir compléter l'architecture orientée comportement à laquelle nous souhaitons cependant laisser le contrôle de l'agent. Nous terminons ce chapitre par la présentation des domaines proches auxquels nous souhaiterions appliquer notre modèle.

Les dernière pages de ce rapport sont consacrées à la conclusion générale de notre travail.

Conclusion générale

Lors de cette thèse nous sommes partis du domaine de l'animation comportementale. Ces recherches se fixent pour objectif de créer des personnages animés, capables de prendre certaines décisions. Les personnages virtuels possèdent de nombreux points communs avec les agents autonomes. Comme eux, ils doivent résoudre le problème général de la sélection d'actions.

Le problème de la sélection d'actions est un problème récurrent et ouvert. Malgré, les progrès scientifiques et techniques, il n'existe pas de solution universellement reconnue, capable de traiter la sélection d'actions dans le cadre d'agents situés dans des environnements dynamiques et continus. Depuis ses débuts, dans la fin des années quatre-vingts l'approche orientée comportement se place dans ce type d'environnements. C'est pour cela qu'elle s'est imposée dans le domaine de la robotique mobile et influence les systèmes multi-agents et l'animation.

Les architectures orientées comportement sont fondées sur l'hypothèse qu'une décision peut être répartie entre différentes entités (les comportements). Chaque comportement étant chargé d'un aspect particulier du problème. L'une des difficultés de cette approche est de dégager une décision globale à partir des suggestions individuelles. Comme le montre le second chapitre, de nombreuses méthodes permettant de coordonner les comportements ont été proposées. Elles sont regroupées en deux principales familles : l'arbitrage et la fusion d'actions. L'arbitrage possède le défaut majeur de ne prendre en compte que la préférence d'un seul des comportements. La fusion d'actions, dont le vote fait partie, essaie de sélectionner une action qui réalise un compromis entre les préférences des comportements. Le vote présente de nombreux avantages. Il permet, en particulier, de sélectionner des actions qui satisfont la majorité des comportements. Cependant, les architectures existantes ne permettent pas de sélectionner des actions dans un domaine continu, utilisent des pondérations qui biaisent la sélection et mettent en oeuvre des comportements de granularité parfois trop élevée.

Nous avons donc proposé, dans le chapitre III, une modification de la méthode de coordination par vote. Afin de prendre en compte les remarques précédemment citées, nous laissons les comportements proposer les options candidates au vote. Nous supprimons les pondérations et proposons d'instancier dynamiquement les comportements. Notre modèle permet de sélectionner les décisions dans un espace continu et d'utiliser des comportements de granularité fine. Il met en oeuvre une procédure de vote fondée sur la méthode de Borda. Dans ce chapitre, nous avons donc décrit les données manipulées par les comportements : veto et options candidates au vote. Puis nous avons présenté notre mécanisme de sélection d'actions, réparti entre les comportements et le mécanisme central de coordination. Nous avons dégagé six étapes majeures qui permettent de sélectionner une action.

Nous avons appliqué le modèle décisionnel à la navigation réactive autonome en environnement virtuel, lors du chapitre IV. La navigation est un problème intéressant et ouvert. Les essais ont été réalisés en environnement statique et dynamique. Ils prouvent l'intérêt du modèle et n'utilisent aucune pondération. Les résultats en environnement statique mettent en avant les avantages de notre modèle pour la coordination des comportements et pour la persistance dans ces choix. Par rapport aux autres architectures orientées comportement, notre modèle est capable d'atteindre son objectif en respectant des contraintes. Les tests en environnement multi-agents montrent l'apparition d'un phénomène émergent (évitement d'un groupe d'agents). Ils montrent également les limites du modèle de navigation. Elles se traduisent, en particulier, par l'apparition de blocages. C'est pourquoi, les résultats réalisés dans un environnement dynamique très contraint décroissent lorsque le nombre d'agents est supérieur à vingt.

Le cinquième chapitre présente les perspectives liées à notre contribution. De par le caractère ouvert du problème traité, ce travail fait l'objet de plusieurs perspectives théoriques et applicatives. En effet, nous n'avons pas traité tous les problèmes auxquels les piétons virtuels sont confrontés (choix d'un endroit pour traverser, navigation en groupe, interaction avec l'avatar d'un joueur, etc.). Ces perspectives applicatives présentent des cadres intéressants pour des recherches théoriques sur les systèmes multi-agents (interactions entre groupes d'agents, systèmes hétérogènes, etc.). En outre, nous envisageons d'améliorer (comportement inertie, réduction du nombre d'options, etc) et d'étendre notre modèle décisionnel (liaison avec un module de niveau d'abstraction supérieur).

Bibliographie

[Adam, 2000] E. Adam, "Modèle d'organisation multi-agent pour l'aide au travail coopératif dans les processus d'entreprise, application aux systèmes administratifs complexes". Thèse de l'université de Valenciennes, janvier 2000.

[Airault et al., 2004] V. Airault, S. Espié, C. Lattaud, J-M. Auberlet, "Interaction betwenn pedestrians and their environment when road-crossing: A behavioural approach", Chioggia, Italy - 27-29 October 2004.

[Arkin, 1989] R. C. Arkin, "Motor Schema-Based Mobile Robot Navigation", The International Journal of Robotics Research, vol. 8, n°4, 1989, pp. 92-112.

[Arkin, 1998] R. C. Arkin, "Behavior-based Robotics", The MIT Press, Cambridge, Massachusetts, USA, 1998.

[Arnaud, 2000] P. Arnaud, "Des Moutons et des robots", Presses polytechniques et universitaires romandes, Lausanne, 2000.

[Arrow, 1951] K. J. Arrow, "Social choice et individual values", Wiley, New York, 1951.

[Aubert et al., 2003] G. Aubert, C. Charron, S. Jung, M-A. Granié, E. Grislin-Le Strugeon, F-X. Lepoutre, P. Pudlo, "Rapport final du projet PREDIT RESPECT", LAMIH, Valenciennes, janvier, 2003.

[Aylett et al., 2000] R. Aylett, M. Luck, "Applying Artificial Intelligence to Virtual Reality: Intelligent virtual environments", Applied Artificial Intelligence, volume 14, n°1, janvier 2000.

[Blach et al., 1995] T. Balch, G. Boone, T. Collins, H. Forbes, D. MacKenzie, and J. Santamaria, "Io, Ganymede and Callisto : A Multiagent Robot Trash-Collecting Team", AI Magazine, 16(2): pp. 39-53, 1995.

[Blumberg et al., 1995] B. M. Blumberg, T. A. Galyean, "Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environment", in proceedings of SIGGRAPH 95, Addison-Wesley. Los Angeles, California, 6-11 august 1995

[Bond et al., 1988] A H. Bond, L Gasser (Eds.), Readings in Distributed Artificial Intelligence, Morgan Kaufmann, 1988.

[Borda, 1781] J-C. De Borda, "Mémoire sur les élections au scrutin", Histoire de l'Académie Royale des Sciences, Académie Royale des Sciences, Paris, 1781.

[Boursin, 1990] J. L. Boursin, "Les dés et les urnes. Les calculs de la démocratie", Seuil, Paris, 1990.

[Briot et al., 2001] J-P. Briot et Y. Demazeau (sous la direction de), "Principes et architectures des systèmes multi-agents", Hermes Science et Publications, Paris, 2001.

[Brooks, 1986] R. A. Brooks, "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, RA-2/1, pp. 14-23, 1986.

[Brooks et al., 1998] R.A. Brooks, C. Breazeal (Ferrell), R. Irie, C. Kemp, M. Marjanovic, B. Scassellati and M. Williamson, "Alternate Essences of Intelligence", Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, pp. 961-976, July 1998.

[Bryson, 2002] J. Bryson,"The Behavior-Oriented Design of Modular Agent Intelligence", In the Proceedings of Agent Technology and Software Engineering (AgeS 02), Erfurt, Allemagne 8 et 9 octobre 2002.

[Caicédo et al., 2000] A. Caicedo, D. Thalmann, "Virtual Humanoids: Let be autonomous without losing control". Proceedings of 4th International Conference Computer Graphics and Artificial Intelligence (3IA'2002), Limoges, France, 2000.

[Carabela et al., 2003] C. Carabela, O. Boissier, A. Florea, "Autonomie dans les systèmes multi-agents. Essai de classification", Actes de JFSMA 2003, Journées Francophones sur les systèmes Multi-agents (27-29 novembre 2006, Hammamet, Tunisie), Revue des Sciences et Technologies de l'Information, Hermes, Lavoisier, Paris, 2006.

[Castelfranchi, 1995] C. Castelfranchi, "Guarantees for Autonomy in Cognitive Agent Architecture" In Intelligent Agents: ECAI-94 Workshop on Agents Theories, Architectures, and Languages, Wooldridge, M. J. and Jennings, N. R., (eds.), 1995, Springer-Verlag, Berlin, 1995.

[Chaib-draa, 1996] B. Chaib-draa, "A Hierarchical Model of Agent Based on Skill, Rules, and Knowledge", In Advances in Artificial Intelligence, 11th Biennal Conf. on AI, Toronto, Canada, Springer Verlag, LNAI 1081, McCalla (Ed.), Berlin, 1996.

[Champion, 2003] A. Champion, "Mécanisme de coordination multi-agents fondé sur des jeux : Application à la simulation comportementale de trafic routier en situation de carrefour", thèse de l'université de Valenciennes, décembre 2003.

[Chopra-Khullar et al., 1999] S. Chopra-Khullar, N. Badler, "Where to look? Automating attending behaviors of virtual human characters". In Proceedings of ACM Autonomous Agents'99, Seattle, USA, May 1999.

[Condorcet, 1785] N. De Condorcet, "Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix", Imprimerie royale, 1785.

[Dalgalarrondo, 2001] A. Dalgalarrondo, "Intégration de la fonction perception dans une architecture de contrôle de robot mobile autonome", Thèse de l'université de Paris-Sud, Centre d'Orsay, janvier 2001.

[Dalgalarrondo, 2003] A. Dalgalarrondo , "À propos de l'autonomie des robots", rapport n° CTA / 02 350 108 / RIEX / 807, juillet 2003, DGA / DCE / Centre Technique d'Arcueil.

[Decugis et al., 1998] V. Decugis et J. Ferber, "An extension of Maes' action selection mechanism for animats" in R. Pfeifer, B. Blumberg, J.-A. Meyer, & S.W. Wilson (Eds.) From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptative Behavior (pp. 153-158). Cambridge, Massachusetts : MIT Press, 1998.

[Demazeau, 1995] Y. Demazeau, "From interactions to collective behaviour in multi-agents systems", First European Conference on Cognitive Science, Saint-Malo, France, 1995.

[Demazeau, 2003] Y. Demazeau, "Créativité emergente centrée utilisateur", dans J-P.Briot and K.Ghadira (eds.): Déploiement des SMA – vers un passage à l'échelle, Actes de JFSMA 2003, Journées Francophones sur les systèmes Multi-agents (27-29 novembre 2006, Hammamet, Tunisie), Revue des Sciences et Technologies de l'Information, Hermes, Lavoisier, Paris, 2006.

[Demazeau, 2005] Y. Demazeau "Multi-Agent Systems Featuring Trust and User Centering", in M. Mohammadian (Ed.), Proceedings CIMCA'2005 International Conference on Computational Intelligence for Modeling, Control & Automation, Vienna, Austria, 28-30 November 2005.

[Devillers, 2001] F. Devillers, "Langage de scenario pour des acteurs semi-autonomes", thèse de l'université de Rennes 1, septembre 2001.

[Doniec et al., 2006] A. Doniec, R. Mandiau, S. Piechowiak, S. Espié, "L'anticipation comme modèle d'interaction : application à la coordination multi-agent en simulation", actes de RFIA 2006 15ème congrès francophone Reconnaissance des formes et Intelligence Artificielle (25-27 janvier 2006, Tours), Presses Universitaires François-Rabelais, Tours, janvier, 2006.

[Dorer, 1999] K. Dorer, "Behavior networks for continuous domains using situation–dependent motivations", Proceedings of the Sixteenth International Conference of Artificial Intelligence, pp 1233–1238, Las Vegas, Nevada, USA, juin 1999.

[Drogoul, 1993] A. Drogoul , "De la simulation Multi-agents à la résolution collective de problèmes", thèse de l'université de Paris VI, novembre 1993

[Duthen, 2002] Y. Duthen, "Behavioural Simulation and Artificial Life", In D. Plemenos (Ed.), Proceedings of 5th International Conference Computer Graphics and Artificial Intelligence (3IA'2002), Limoges, France, Diazo1, Clermont-Ferrand, pp. 5-6, janvier, 2002

[Ebel et al., 2002] A. Ebel, D. Hanon, B. Stanciulescu, P. Pudlo, E. Grislin-le strugeon , F-X. Lepoutre, "Modèle d'animation comportementale de piétons virtuels", XVèmes Journées de l'Association Française d'Informatique Graphique, Lyon, France, décembre, 2002.

[Erman, 1980] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty", ACM Computing Survey 12:213-253, 1980.

[Farenc et al., 1998] N. Farenc, S. Raupp-musse, E. Schweiss, M. Kallman, O. Aune, R. Boulic, D. Thalmann, "One Step Towards Virtual Human Management for Urban Environment Simulation", In Proceedings of the ECAI Workshop on Intelligent User Interfaces, Brighton, United Kingdom, 1998.

[Ferber, 1995] J. Ferber, "Les systèmes multi-agents : vers une intelligence collective", InterEditions, Paris, 1995.

[Fergusson, 1992] I. A. Fergusson, "TouringMachines: an architecture for dynamic, rational, mobile agents", Technical Report n°273 Univertity of Cambridge, November 1992.

[Flacher et al., 2003] F. Flacher , O. Sigaud , "Coordination spatiale émergente par champs de potentiels", Numéro spécial de la revue Technique et Science Informatiques : Vie artificielle, Guillot A. et J-A Meyer (eds), Hermes, Paris, pp. 171-195, 2003.

[Fluchs et al., 2003] P. Fluchs et G. Moreau (sous la direction de), "Le Traité de la réalité virtuelle, 2ème édition, volume 1 et 2", Les Presses de l'Ecole des Mines, Paris, 2003.

[Fruin, 1971] J. Fruin, "Pedestrian and Planning Design", Elevator World Inc, Alabama, 1971. (Cité dans [Lamarche, 2003])

[Girard et al., 2001] B. Girard, G. Robert, A. Guillot, "Jeu Vidéo et Intelligence Artificielle Située", in Cognito. 22, 57-72, 2001.

[Girard, 2003] B. Girard,"Intégration de la navigation et de la sélection de l'action dans une architecture de contrôle inspirée des ganglions de la base", thèse de l'université Paris 6, Septembre 2003.

[Goffman, 1971] E. Goffman . "Relations in public : microstudies of the public order", Basic Books, New York, 1971. (cité dans [Tomas, 1999])

[Granié et al., 2001] M.A. Granié, G. Aubert, "Projet RESPECT : Dossier de spécifications, volume-1 : Scénarios d'accidents d'enfants piétons", 22 novembre 2001.

[Grislin et al., 1996] E. Grislin-Le Strugeon, G. Agimont, R. Mandiau, P. Millot, "Organisation évolutive d'un système multi-agents. Illustration par des agents-robots miniers", In Müller J.p. Et Quinquenton J. (Ed.), I.A. Distribuée et systèmes multi-agents, Hermès, Paris, pp. 167-176, janvier 1996.

[Grislin et al., 2001] E. Grislin-Le Strugeon, E. Adam, C. Kolski, "Agents intelligents en interaction Homme-Machine dans les Systèmes d'information." dans : "Environnements évolués et évaluation de l'I.H.M., Interaction Homme-Machine pour les Systèmes d'Information", C. Kolski (ed.), Hermès, 2001.

[Grislin et al., 2002] E. Grislin-Le Strugeon, D. Hanon, P. Pudlo, F-X. Lepoutre," Agent-based modelling of autonomous virtual humans". In D. Plemenos (Ed.), Proceedings of 5th International Conference Computer Graphics and Artificial Intelligence (3IA'2002), Limoges, France, Diazo1, Clermont-Ferrand, pp. 170-174, janvier, 2002.

[Grislin et al., 2005] E. Grislin-le strugeon, D. Hanon, R. Mandiau, "Behavioral Self-control of Agent-Based Virtual Pedestrians", In F.f. Ramos, V. Larios Rosillo, H. Unger (Ed.), Advanced Distributed Systems: 5th International School and Symposium, ISSADS 2005, Guadalajara, Mexico, January 24-28,2005, Revised Selected Papers, Lecture Notes in Computer Science, Springer, pp. 529-537, 2005.

[Guillot, 1999] A. Guillot, "Pour une approche dynamique des animats.", Dans A. Drogoul, et J.-A. Meyer, (eds), Intelligence Artificielle Située, Hermès, Paris, pp. 33-58, 1999.

[Gutknecht et al,1998] O. Gutknecht, J. Ferber, "Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents". Journée Francophones IA Distribuée et SMA, Nancy, Hermès, pp. 267-280, 1998.

[Hanon et al., 2002] D. Hanon, E. Grislin-le Strugeon, "Modélisation de comportement du piéton virtuel", Dossier de spécification et de conception, projet RESPECT, Juillet, 2002

[Hanon et al., 2003] D. Hanon, E. Grislin-le Strugeon , R. Mandiau, "A behavior based architecture for the control of virtual pedestrians", In Prahlad Vadakkepat, Tan Woei Wan, Tan Kay Chen, Loh Ai Poh (Ed.), The second International Conference on Computational Intelligence, Robotics and Autonomous Systems CIRAS 2003 (15-18 december, Singapore), CIC, National University of Singapore, Singapore, pp. 125-132, décembre. 2003.

[Hanon et al., 2006] D. Hanon, E. Grislin-le Strugeon, R. Mandiau, "Un modèle décisionnel orienté comportement utilisant le vote, application à la navigation autonome en environnement simulé", dans RFIA 2006 15ème congrès francophone Reconnaissance des formes et Intelligence Artificielle (25-27 janvier 2006, Tours), Presses Universitaires François-Rabelais, Tours, janvier 2006.

[Harnad, 1990] S. Harnad, "The Symbol Grounding Problem", Physica D : Nonlinear Phenomena, volume 42, pp. 335 - 346, 1990.

[Helbling et al., 2000] D. Helbling, I. Farkas, T. Vicsek , "Simulating dynamical features of escape panic", Nature n°407, Septembre 2000.

[Hostetler et al., 2002] Hostetler T., Karrney J., "Strolling Down the Avenue with a Few Close Friends", Eurographics Ireland 2002 Workshop Proceedings, Dublin Ireland, pp. 7-14, Mars 2002.

[Hostetler, 2003] T. Hostelter, "From Flocking to Walking : Simulating Groups of Virtual Characters", séminaire du LAMIH, Université de Valenciennes, juin 2003.

[Hudry, 2003] O. Hudry, "Votes et paradoxes : les élections ne sont pas monotones !", Revue Mathématiques et Sciences humaines, n° 163, pp. 9-39, 2003.

[Isla et al., 2001] D. Isla, R. Burke, M. Downie, B. Blumberg, "A Layered Brain Architecture for Synthetic Creatures", International Joint Conferences on Artificial Intelligence (IJCAI), Seattle, USA, pp. 1051-1058, August 2001.

[Jørgensen et al., 2004] M. W. Jørgensen, E. H. Østergaard and H. H. Lund, "Modules for a self-reconfigurable robot", Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 28 – October 2, Sendai, Japan, 2004.

[Kallmann et al., 2000] M. Kallmann, J.S. Monzani, A. Caicedo, D. Thalmann, "ACE : A platform for the real Time Simulation of Virtual Human Agents", Proceedings of the 11th Eurographics Work on Animation and Simulation 2000, Interlaken, Switzerland, August 2000.

[Kennedy et al., 2001] J. Kennedy, R. C. Eberhart, Y. Shi, "Swarm intelligence", Morgan Kaufmann Publishers, San Francisco, 2001.

[Labidi et al., 1993] S. Labidi, W. Lejouad, "De l'Intelligence Artificielle Distribuée aux Systèmes Multi-Agents", Rapport de recherche de l'INRIA n°2004, Août 1993.

[Lamarche, 2003] F. Lamarche, "Humanoïdes virtuels, réaction et cognition : une architecture pour leur autonomie", thèse de l'université de Rennes 1, décembre 2003.

[Le Strugeon, 1995] E. Le Strugeon, "Une méthodologie d'auto-adaptation d'un système multi-agents cognitifs". Thèse de l'université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, janvier 1995.

[Maes, 1989] P. Maes, "The dynamics of action selection", Proceedings of the International Joint conference on Artificial Intelligence, IJCAI-89,1989.

[Mandiau, 1993] R. Mandiau, "Contribution à la modélisation des univers multi-agents : génération d'un plan partagé", thèse de l'université de Valenciennes et du Hainaut-Cambrésis, janvier 1993.

[Mandiau et al., 2002] R. Mandiau, E. Grislin-Le Strugeon, A. Péninou (sous la direction de), "Organisation et applications des SMA". Hermès, Paris, 2002.

[Mataric, 1994] M. J Mataric,"Interaction and Intelligent Behavior", thèse du Massachusetts Institute of Technology, mai 1994.

[Mathieu et al., 2003] P. Mathieu , S. Picault , J-C. Routier, "Simulation de comportements pour agents rationnels situés" dans "Modèles formels de l'interaction, actes des secondes Journées Francophones", A. Herzig, B. Chaib-draa, P. Mathieu (eds.), Cépaduès-éditions, Lille, mai 2003.

[Minsky, 1988] M. Minsky, "La société de l'esprit", InterEditions, Paris, 1988.

[Moreau, 1998] G. Moreau , "Modélisation du comportement pour la simulation interactive application au trafic routier multimodale", thèse de l'université de Rennes 1, novembre 1998.

[Müller, 1996] J.P. Müller "The Desing of intelligent agents, A layered Approach", Lecture Notes in AI, vol. 1177, Springer-Verlag, 1996.

[Müller, 1997] J. P. Müller, "Control Architecture for Autonomous and Interacting Agent : A Survey", In "Intelligent Agent Systems, Theorical and Practicall Issues", Lecture Notes in AI 1209, LN in CS, Springer, 1997.

[Noser et al., 1995] H. Noser, D. Thalmann, "Synthetic vision and audition for digital actors", In Proceeding of Eurographics'95, pp 325-336, 1995.

[Parker, 1994] L. E. Parker, "Heterogeneous Multi-Robot Cooperation, thèse du Massachusetts Institute of Technology, January 1994.

[Perlin, et al., 1996] K. Perlin, A. Goldberg, "Improv : A system for scripting Interactive Actor In Virtual Worlds", Computer Graphics proceedings, pp 205-216, 1996.

[Pettre, 2002] J. Pettre, "Planification de marche pour acteur virtuel", Rapport LAAS (Laboratoire d'analyse et Architecture des Systèmes) n°02179, mai 2002.

[Pina et al., 2000] A. Pina, E. Cerezo, F.J. Seron, "Computer animation: from avatars to unrestricted autonomous actors (A survey on replication and modelling mechanisms)", Computer & Graphics, vol. 24, pp. 297-311, 2000.

[Pirjanian, 1998] P. Pirjanian, "Multiple Objective Action Selection & Behavior Fusion using Voting", thèse de l'université d'Aalborg, aout 1998.

[Pirjanian, 1999] P. Pirjanian, "Behavior Coordination Mechanisms – State-of-the-art", Research Report Robotics Research Laboratory, University of Southern California, October 1999.

[Pylyshyn, 1987] Pylyshyn, Zenon W. (Ed.) "The Robot's Dilemma: The Frame Problem in Artificial Intelligence". Norwood, NJ: Ablex Publishing Corporation. 1987.

[Querrec et al , 2001] R. Querrec, P. De Loor, P. Chevalier, "Environnement virtuels pour la formation des officiers sapeurs-pompier". Actes de JFIADSMA'2001, Journées Francophones I. A. Distribuée et systèmes multi-agents, Hermes, Paris, 2001.

[Querrec, 2002] R. Querrec, "Les Systèmes multi-Agents pour les Environnements Virtuels de Formation". Thèse de l'université de Bretagne Occidentale, 4 octobre 2002.

[Rao et al., 1991] A. S. Rao, M. P. Georgeff, "Modeling rational agents whithin a BDI architecture", Proceedings of KR Reasonning, pp 473-484, 1991.

[Rasmussen, 1983] J. Rasmussen, "Skills, rules, knowledge ; signals, signs, and symbols, and other distinctions in human performance models", IEEE Transactions on Systems, Man and Cybernetics, 13, pp. 257-266, 1983.

[Reichgelt, 1990] H. Reichgelt, "Differents styles of agents architectures", Proceedings of the 1st belief representation and agent architectures workshop, Gallier J.R. (ed), pp29-39, Cambridge, UK, mai 1990.

[Reynolds, 1987] C.W. Reynolds, "Flocks, Herd and shools : A distributed behavioural model". Computer Graphics, Vol 21, 1987.

[Reynolds, 1999] C.W. Reynolds, "Steering Behaviors For Autonomous Characters", in the proceedings of Game Developers Conference 1999, San Jose, California. Miller Freeman Game Group, San Francisco, California. pp. 763-782, 1999.

[Reynolds, 2000] C. W. Reynolds, "Interaction with Groups of Autonomous Characters", in the proceedings of Game Developers Conference 2000, CMP Game Media Group, San Francisco, California, pp. 449-460, 2000.

[Rezzonico et al., 1995] S. Rezzonico, Z. Huang, R. Boulic, N. Magnenat Thalmann, D. Thalmann, "Consistent Grasping Interactions with Virtual Actors Based on the Multi-sensor HandModel", In prooceding of Virtual Environments '95, Springer, Wien, pp 107-118, 1995.

[Richard, 2001] N. Richard, "Description de comportements d'agents autonomes évoluant dans des mondes virtuels", thèse de l'école nationale supérieure des télécommunications, 2001.

[Robert, 2001] G. Robert , "Rapport d'avancement des travaux", thèse Cifre Société Nevrax - LIP6, décembre 2001.

[Robert, 2005] G. Robert, "MHiCS, une architecture de sélection de l'action Motivationnelle et Hiérarchique à Systèmes de Classeurs pour Personnages Non Joueurs adaptatifs". Thèse de l'université de Paris VI, 2005.

[Rosenblatt et al., 1989] J.K. Rosenblatt and D.W. Payton, "A fine-Grained Alternative to the subsumption Architecture for Mobile Robot Control", proceedings of the IEEE/INNS International Joint Conference on Neural Networks , Washington DC, June 1989, vol. 2, pp. 317-324.

[Rosenblatt, 1996] J.K. Rosenblatt, "DAMN : A Distributed Architecture for Mobile Navigation" Thèse de l'université de Carnegie Mellon, Pittsburgh, septembre 1996.

[Rosenblatt, 2000] J. K. Rosenblatt, "Optimal Selection of Uncertain Actions by Maximizing Expected Utility", Autonomous Robots val 9, no.1, pp17-25, 2000, Kluwer Academic Publishers

[Rosenblatt et al., 2002] J. K. Rosenblatt, S. Williams, H. Durrant-Whyte, "A Behavior-Based Architecture for Autonomous Underwater Exploration". International Journal of Information Sciences, vol 145, no 1-2, pp 69-87, september 2002.

[Rosenschein, 1985] J. S. Rosenschein, "Rational interaction : coopération among intelligent agents." Thèse de l'université de Standford, 1985.

[Rozé, 2003] C. Rozé, "Organisation multi-agents au service de la personnalisation de l'information", thèse de l'université de Valenciennes, décembre 2003.

[Russell et al., 1995] S. Russell, P. Norvig, "Artificial Intelligence: a Modern Approach", Prentice-Hall, 1995.

[Sansonnet et al., 2006] J-P. Sansonnet, D. Leray, J-C. Martin, "Architecture of a Framework for Generic Assisting Conversational Agents", short presentation at Intelligent Virtual Agents Conference, in LNAI 4133, Marina del Rey (CA), pp 145-156, August, 2006

[Scheutz, 2004] M. Scheutz et V. Andronche, "Architectural Mechanisms for Dynamics Changes of Behavior-Based Systems", Transaction on Man, Systems and Cybernetics, 2004.

[Simonin, 2001] O. Simonin, "Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique", thèse de l'université de Montpellier 2, Décembre 2001.

[Steels, 1995] L. Steels, "When are robots intelligent autonomous agent" in *Robotics and Autonomous Systems*, vol. 15, n°1-2, Amsterdam: Elsevier Science Publishers (North-Holland), pp. 3-9, 1995.

[Sukthankar, 1997] R. Sukthankar, "Situation Awareness for Tactical Driving", Thèse du Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, January, 1997.

[Sukthankar et al., 1998] R. Sukthankar, B. Shumeet, "Multiple Adaptative agent for Tactical Driving", International Journal of Articial Intelligence, 9, pp. 1-20, 1998.

[Tecchia et al., 2000] F. Tecchia, Y. Chrysanthou, "Real time Visualisation of Densely Populated Urban Environments : a simple and fast algorithm for collision Detection", Eurographics, Swansee, UK, April 2000.

[Terzopoulos et al., 1994] D. Terzopoulos, X. Tu, R. Grzeszczuk,, "Artificial fishes : Autonomous locomotion, perception, behavior, and learning in a simulated physical world," in Artificial Life, 1, 4, pp. 327-351, December, 1994.

[Thalmann, 2003] D. Thalmann, "Concepts and Models for Inhabited Virtual Worlds", First International Workshop on Language Understanding and Agents for Real World Interaction, Hokkaido University, Sapporo, Japan, July 13, 2003.

[Thomas, 1999] G. Thomas, "Environnements virtuels urbains: modélisation des informations nécessaires à la simulation de piétons", thèse de l'université de Rennes 1, décembre 1999.

[Tisseau, 2001] J. Tisseau, "Réalitée Virtuelle, *autonomie in virtuo*" ,Habilitation à Diriger des Recherches de l'université de Rennes 1, décembre 2001.

[Tyrrell, 1993] T. Tyrrell, "Computational Mechanisms for Action Selection", thèse de l'université d'Edimbourg, 1993.

[Wooldridge et al., 1995] M. Wooldridge, N. R. Jennings, "Intelligent Agents : Theory and Practice". In Knowledge Engineering Review N°10, 1995.

[Wooldridge et al., 1998] M. Wooldridge, N. R. Jennings, "Pitfalls of Agent-Oriented Development", In K. P. Sycara and M. Wooldridge (eds.), proceedings of the Second International Conference on Autonomous Agents ACM Press, May 1998.

[Zeghal, 1993] K. Zeghal, "Champs de Forces Symétriques : un Modèle de Coordination d'Actions Réactive Appliqué au trafic Aérien", Rapport LAFORIA n° 93/14, mai 1993.

Résumé

Les acteurs virtuels autonomes sont des agents évoluant dans des environnements dynamiques et continus. Ils doivent prendre des décisions afin de s'y adapter en temps réel. Il leur faut donc disposer de modèles décisionnels performants. Les modèles décisionnels orientés comportement sont particulièrement adaptés à ces problèmes. Ils sont fondés sur l'hypothèse qu'une décision peut être répartie entre différentes entités (ou comportements). L'une des difficultés de cette approche est de dégager une décision globale à partir des suggestions des comportements. Dans ce cadre, le vote présente de nombreux avantages. Cependant, dans la littérature, il ne permet pas de sélectionner des actions continues et utilise des pondérations qui biaisent la sélection.

Notre contribution consiste en une modification de la méthode de coordination par vote. Notre proposition permet de sélectionner les décisions dans un espace continu, d'utiliser des comportements de granularité fine et met en oeuvre une procédure de vote différente. Nous avons appliqué le modèle décisionnel à la navigation réactive autonome en environnement virtuel. Les essais réalisés prouvent l'intérêt du modèle et n'utilisent aucune pondération. Les résultats en environnement statique montrent que le modèle est capable d'atteindre son objectif en respectant des contraintes et en persistant dans ses choix. Les résultats en environnement dynamique décroissent lorsque le nombre d'agents est supérieur à vingt.

De par le caractère ouvert du problème traité, ce travail fait l'objet de plusieurs perspectives applicatives (simulation de groupes d'agents) et théoriques (liaison avec un module de niveau d'abstraction supérieur).

Mots Clefs : Systèmes multi-agents, Modèle décisionnel, Architecture orientée comportement, Vote, Navigation.