Modèles et simulations informatiques des problèmes de coopération entre agents

Bruno Beaufils

Modèles et simulations informatiques des problèmes de coopération entre agents

THÈSE

présentée et soutenue publiquement le 25 janvier 2000

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille

(spécialité informatique)

par

Bruno Beaufils

Composition du jury

Président :	Pr. Jean-Marc Geib	LIFL, Université des Sciences et Technologies de Lille
Rapporteurs :	Pr. Jean-Gabriel Ganascia	LIP6, Université Pierre et Marie Curie
	Pr. Marc Schoenauer	CMAP, École Polytechnique
Examinateur :	Pr. Philippe Preux	LIL, Université du Littoral - Côte d'Opale
Directeurs :	Pr. Jean-Paul Delahaye	LIFL, Université des Sciences et Technologies de Lille
	Pr. Philippe Mathieu	LIFL, Université des Sciences et Technologies de Lille

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
Laboratoire d'Informatique Fondamentale de Lille — UPRESA 8022
U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. : +33 (0)3 20 43 47 24 – Télécopie : +33 (0)3 20 43 65 66 – email : direction@lifl.fr

Mis en page et composé avec LaTeX et la classe thlifl.

À mes parents et grand-parents.

Remerciements

« *La page des remerciements est bien souvent la dernière écrite d'une thèse (celle-ci n'échappant pas à la règle), mais également la première (et souvent la seule) lue* ». C'est la phrase d'une des thèses soutenues au LIFL ces dernières années dont je me souviens le mieux [1]. Je vais donc essayer de n'oublier personne dans les indispensables remerciements que je me dois de faire, tant il est vrai qu'une thèse est loin d'être un travail solitaire.

Tout d'abord je ne peux exprimer ici toute ma gratitude envers mes deux directeurs de thèse, Jean-Paul Delahaye et Philippe Mathieu. J'ai pu apprécier au cours des années passées à leur côté leur passion communicative et leur curiosité sans limites non seulement pour l'informatique mais aussi plus largement pour les sciences et la rationalité. Les discussions interminables que nous avons eues m'ont permis de garder un esprit ouvert et critique bien au delà du domaine de la thèse. Ils ont su m'initier aux domaines fascinants de l'intelligence artificielle et de la vie artificielle et ce bien avant la thèse. Ils m'ont fait confiance très tôt, tout en sachant m'indiquer les directions à suivre, et me montrer, avec insistance, mes défauts méthodologiques. Je veux donc ici les remercier en premier. J'espère simplement que le travail que j'ai fourni n'a pas trahi leur confiance ni leurs espérances et me permettra un jour de gagner un pari contre Philippe.

Je tiens à remercier les membres du jury de l'honneur qu'ils me font en ayant accepté de s'intéresser à mon travail. Merci à Jean-Gabriel Ganascia et Marc Schoenauer d'avoir rapporté ce travail dans des conditions si rapides et malgré leurs responsabilités et leur emploi du temps si chargé. Leur réputation scientifique m'a occasionné quelques frayeurs pendant la période d'attente de leur rapports et l'honneur qu'ils me font en rapportant ce travail n'en est que plus grand. Merci à Jean-Marc Geib d'avoir accepté de présider ce jury. Notre équipe de recherche ne peut que se féliciter d'avoir été intégrée à l'axe CIM qu'il dirige et dont il a réussi à assurer la très forte cohésion. Merci enfin à Philippe Preux, qui a été le premier à me communiquer le virus des phénomènes évolutifs lors de séminaires de DEA et dont l'ouverture d'esprit scientifique est aussi un modèle pour moi.

Je remercie bien entendu mes parents, sans le soutien et la présence desquels, je n'aurais jamais pu envisager de faire ce travail, et à qui je dédie cette thèse ; mais aussi ma sœur et Stéphane, dont je suis assez fier de l'originalité de la rencontre.

Je tiens ensuite bien sûr à remercier l'ensemble des nombreux membres du bureau 318, passés, passants ou présents, je remercie plus particulièrement Stéphane, Jean-Marc, Hélène, Tof, Ramón, Olivier et Zébulon. Je tiens à remercier séparément Sébastien et Jean-Stéphane qui ont été bien plus que des camarades de jeux forts compréhensifs. Je ne pouvais rêver d'une ambiance de travail plus agréable. Je vous remercie tous pour les nombreuses discussions, parties de flippers et autres dégustations qui ont si subtilement fait progresser notre travail et nos goûts musicaux à tous.

Plus généralement je veux remercier tous les membres de SMAC, de CIM, et du LIFL, qui ont fait de la vie de tous les jours un moment agréable. J'ai une pensée particulière pour ceux qui ont subi mes harcèlements quasi-quotidiens, des grilmens (notamment Sam, Gilles, Luc et Laurent) aux personnels administratifs (notamment Annie, Patricia et Bruno).

Je ne peux évidemment pas oublier les membres de la famille Schtroll qui ont toujours été là pour m'aider, me divertir ou me remettre en place. Ils sont trop nombreux pour être tous cités mais ils se reconnaîtront facilement, des grands-parents aux cousins éloignés. Merci à vous d'avoir été dans mes racines et de rester aussi bizarres, originaux et donc *extra-ordinaires* les uns que les autres.

Merci aux moniteurs du groupe des 18, plus particulièrement Carine, Sandrine et Laurent dont la rencontre aura été finalement plus enrichissante sinon formatrice que les institutions ne le prévoyaient.

Pour finir je tiens à adresser les remerciements les plus chaleureux à Sonia qui, en plus de sa gentillesse, a du supporter mon caractère, disons original, notamment lors de la rédaction de nos thèses respectives. Sa présence à mes côtés est une joie chaque jour renouvellée.

1. L'auteur se reconnaîtra et, j'en suis sûr, ne m'en voudra pas de ne pas avoir retenu d'abord l'ensemble de ces résultats si nombreux.

Table des matières

Liste des tableaux

Table des figures

Introduction

Motivations

En Vie Artificielle, la modélisation d'organismes primitifs est la base d'études. La vie prend forme dans des mémoires de silicium de la rencontre de centaines de tels agents. Les interactions entre ceux-ci permettent de faire émerger des comportements complexes et non prévisibles. C'est ainsi que l'on espère comprendre les mécanismes de la vie telle qu'elle est [Lan89].

De manière théorique, on définit alors les unités de base de ces systèmes vivants comme des agents : des entités autonomes, ayant un but précis. Les différents mondes que l'on représente sont alors des systèmes contenant un certain nombre de ces agents. On dit souvent que ces mondes sont des systèmes multi-agents, [Fer95].

Les agents de la Vie Artificielle sont la plupart du temps des représentations informatiques simplifiées d'êtres vivants. La Vie Artificielle se veut être une sorte de biologie informatique expérimentale. Les êtres vivants réels, que ce soient les humains, les animaux, ou les bactéries, semblent avoir un but principal commun : la survie. La vie est un mécanisme dynamique qui s'auto-entretient. Quand ils partagent le même environnement, les agents sont alors en compétition pour l'acquisition, par exemple, de nourriture, ou d'espace. Les représentations informatiques de ces mondes doivent donc prendre en compte le comportement que les entités modélisées peuvent avoir. En simplifiant, on arrive à deux comportements de base la coopération et la non-coopération.

Robert Axelrod, dans un ouvrage qui fait depuis référence, [Axe84], s'est interessé à la modélisation des différents comportements basé sur ces deux actes de base. Son modèle d'étude est issue de la théorie des jeux de von Neumann et Morgenstern, [vNM44] : le dilemme itéré du prisonnier. Dans le dilemme du prisonnier, deux agents se retrouvent dans la position de faire un choix entre deux actions. S'ils restent rationnels et choisissent l'action qui devrait favoriser leur intérêt individuel, alors ils gagnent moins que s'ils choisissent l'action qui favorise l'ensemble des participants. De là vient le dilemme : de la non coincidence entre intérêt individuel et intérêt collectif.

La théorie des jeux n'apporte en fait que peu de résultats concrets sur les comportements à adopter de façon à favoriser la coopération. La définition de la rationalité utilisée est sans doute le frein majeur à cela. C'est pourquoi de nombreux travaux ont été faits par simulations : des stratégies sont testées de manière virtuelle. Le résultat permet alors de savoir si ces comportements favorisent ou non la coopération.

Robert Axelrod a proposé de simuler par un programme informatique différents types de comportements pour le dilemme du prisonnier. Un très grand nombre de règles lui furent soumises. Grâce à ses simulations, il a réussi à exhiber certains traits de caractères qui semblent importants pour favoriser la coopération. D'après lui, quatre propriétés doivent être regroupées dans un comportement pour que la coopération puissent intervenir dans un grand nombre de cas : la bienveillance, la réactivité, l'indulgence, et la simplicité. Une stratégie regroupe selon lui ces quatres caractéristiques : `tit_for_tat`. Cette stratégie commence par coopérer, puis imite le comportement de son adversaire.

Par la suite, de très nombreux travaux ont semblé confirmer les qualités de cette stratégie. Des expériences paraissaient confirmer ses qualités, [Dav87] ; des observations comparaient le comportement d'agents vivants à cette stratégie. `tit_for_tat` est devenu **la** stratégie de référence pour tout ce qui concerne la coopération entre agents.

Nowak et Sigmund ont mené de très nombreuses expériences et études formelles en utilisant des stratégies stochastiques, [NS89, NS90, Now90]. Lindgren a étudié plus précisément l'évolution des comportements dans le dilemme du prisonnier, [Lin92, LN95]. Les qualités de `tit_for_tat` ont alors commencé à être remises en cause, [NS93].

La simplicité qu'Axelrod considère être une qualité pour les comportements coopératifs ne nous semble pas être suffisament justifiée. Avoir un comportement trop simple peut justement permettre à d'autres agents de savoir l'exploiter. Nous pensons au contraire qu'avoir un comportement complexe permet de mieux appréhender les différentes approches de la coopération. Comme semble le laisser penser l'exemple du cerveau humain, qui est plus volumineux que celui des autres primates, les stratégies coopératives sont à notre avis infiniment complexes, et le nombre, comme la nature des critères de qualité, pour de telles stratégies ne sont pas limités.

Nous essayons donc par des simulations de déterminer de nouvelles qualités pour la coopération. Nous utilisons pour cela des modèles proches de ceux de l'évolution et de la sélection naturelle. Nous adaptons donc la méthodologie d'Axelrod, et étudions le dilemme du prisonnier dans un cas discret. Ces modèles nous permettent de mener de très larges simulations, afin de non seulement trouver ces qualités inconnues, mais aussi et surtout de vérifier la validité des résultats classiques. Nous arrivons, par exemple, à confirmer les résultats de Nowak et Sigmund, dans un cadre de simulation différent du leur.

D'autre part, nous accumulons les arguments indiquant que le modèle du dilemme du prisonnier n'est pas aussi simple que sa définition le laisse supposer, et confirmons de cette manière la qualité toute relative de `tit_for_tat`.

Plan de la thèse

Dans un premier temps, nous allons donc présenter la théorie des jeux, qui est le cadre de définition initial du dilemme itéré du prisonnier. Nous présenterons l'histoire de l'apparition de cette théorie, afin d'en comprendre les motivations. Ensuite nous introduirons de manière simplifiée les différents concepts qu'utilise la théorie. Nous présenterons à cet effet deux exemples de jeu. Chacun d'eux nous permettra d'introduire des concepts différents. Nous pourrons alors définir de manière précise les termes, vocabulaires et notations de la théorie des jeux, qui seront utilisés dans le reste du document et tels qu'ils sont introduits dans [vNM44, LR85].

Nous présenterons également de manière rapide, sous la forme de brefs rappels, les outils plus précisément informatiques qui nous seront utiles lors de la mise en place des simulations. Cela concerne principalement le concept des automates d'états finis.

Dans le second chapitre nous pourrons alors commencer à présenter le modèle de base d'étude de la coopération : le dilemme du prisonnier. Cette présentation se fera en restant dans son cadre initial : la théorie des jeux.

Nous verrons tout d'abord comment et pourquoi ce modèle est apparu. Ceci se fera notamment par la description commentée des expériences des deux concepteurs du modèle faites dans [Flo52], et de l'explication de l'origine de la parabole servant de description au jeu. Nous pourrons alors montrer comment la théorie des jeux aborde le dilemme du prisonnier, et comment elle faillit à sa tâche. Nous présenterons les différentes approches utilisées en théorie des jeux pour résoudre le problème posé par le modèle quant à la définition de la rationalité, comme le fait Shubik dans [Shu70].

Nous finirons cette présentation par la description du dilemme itéré du prisonnier, qui semble mieux adapté que le modèle de base à l'étude des situations de coopération. Nous verrons en quoi il apporte une meilleure modélisation.

Dans le troisième chapitre nous décrirons de manière précise quelques comportements et des méthodes d'évaluations de ceux-ci. Cette présentation reprendra les travaux d'Axelrod, [Axe84].

À cet effet, nous préciserons quelque peu la terminologie adaptée à la description des comportements ainsi que les outils qu'elle nécessite. Nous pourrons alors donner des exemples de stratégies en

en proposant un début de taxonomie par trait de caractère constitutif. Nous montrerons à travers la littérature que ces comportements ne sont pas complètement artificiels mais se basent sur des observations précises.

Nous pourrons alors présenter les deux méthodes d'évaluation des stratégies les plus souvent rencontrées, à savoir les tournois et les évolutions écologiques. Pour chacune d'elles, nous spécifierons tout d'abord l'idée générale de la méthode, puis nous préciserons les modifications apportées aux définitions afin de pouvoir utiliser ces méthodes de manière pratique sur ordinateur. Nous décrirons les différences méthodologiques par rapport aux travaux de [Axe84, Smi82]. Ces limitations sont en fait toujours basées sur le même principe, à savoir la transformation de l'étude continue en études discrètes et déterministes. Ces deux dernières contraintes sont deux hypothèses fortes et sont celles qui rendent les travaux de la thèse originaux. Cette présentation sera l'occasion d'exposer en détail les résultats classiques, ainsi que de faire le parallèle entre la coopération et l'évolution de la coopération. Nous résumérons à ce moment les différentes controverses à ce sujet.

Pour clore ce chapitre, nous montrerons formellement la difficulté d'étude du dilemme au travers de deux tentatives de réponses à des questions simples. Ce sera l'occasion d'exhiber un cas d'apparition de la complexité dans l'étude de la coopération.

Dans le quatrième chapitre, nous exposerons quelques unes des nombreuses simulations que nous avons menées.

Nous présenterons rapidement le simulateur logiciel qui est utilisé pour tous les résultats apparaissant dans le document. Ensuite nous verrons en détail une méthode de mesure de la robustesse du dilemme du prisonnier que nous avons mise au point et utilisée afin de vérifier que les simulations avaient un sens. Certains des résultats obtenus avec cette méthode seront exposés.

Nous verrons que grâce aux simulations, les dynamiques dans l'évolution de la coopération ne sont pas toujours monotones et simples. Au contraire, nous décrirons une taxonomie de ces évolutions en cinq classes, dont quatre comportent des oscillations dans la représentation des comportements au cours du temps. Nous affinerons ainsi les résultats de Nowak et Sigmund établis dans [NS89] pour le cas continu. Nous verrons également que ces dynamiques sont souvent très sensibles aux variations sur les conditions initiales de simulations, qui font penser que l'évolution de la coopération est peut-être un mécanisme au *bord du chaos*. Nous ne l'affirmerons cependant pas. Il pourra alors être fait pour la seconde fois référence à la présence de complexité dans l'étude de la coopération.

Nous conclurons ce chapitre en présentant deux stratégies nettement supérieures en qualité à `tit_for_tat`. La supériorité de `tit_for_tat` avait déjà été remise en cause entre autre par Nowak et Sigmund dans [NS93]. Nous montrerons sur quelques exemples de simulations leur supériorité. Nous tenterons d'analyser ces stratégies pour en extraire les traits importants : la gradualité et l'adaptabilité de la rétorsion. Nous remarquerons que ces deux nouveaux traits forment avec les traits déjà connus une hiérarchie de stratégies à l'efficacité croissante. Une référence à la complexité dans cette étude sera de nouveau faite.

Dans le cinquième chapitre nous exposerons les travaux faits sur des simulations de grande ampleur, améliorant les expérimentations antérieures, de [ML96] par exemple.

Ces simulations seront motivées par la volonté de créer de grands espaces de stratégies de manière objective. Nous utiliserons pour cela des structures issues des algorithmes génétiques ([Gol89]) permettant de décrire de manière simple un comportement. Il suffira alors d'utiliser toutes les valeurs possibles pour ces structures afin de définir un grand nombre de stratégies différentes. Nous définirons alors les différentes structures que nous avons mises au point. Elles se basent toutes sur des concepts issus directement de l'informatique, à savoir la gestion de la mémoire, et les automates.

Après avoir décrit les expériences menées, nous étudierons alors les résultats de celles-ci et tenterons de les interpréter. Globalement, ces vastes simulations serviront à évaluer d'une nouvelle manière les stratégies, confirmeront nos doutes sur une des qualités avancées par Axelrod et valideront en partie notre étude faite dans le chapitre précédent des deux nouvelles *bonnes* stratégies. Ceci appor-

tera donc un argument supplémentaire en faveur de la présence de la complexité dans l'étude de la coopération et de son évolution.

Pour conclure nous étudierons un modèle légèrement différent du dilemme itéré du prisonnier. Ce modèle fort peu, pour ne pas dire jamais, étudié dans la littérature le sera de manière formelle et pratique à travers des simulations. Nous l'appelerons le dilemme itéré de l'ascenceur.

Une étude a priori de ce modèle nous montrera que les comportements optimaux dans ce cadre ne sont pas déterministes, mais qu'au contraire, font grandement appel au hasard. Ces résultats seront confirmés par des simulations. Une fois de plus, nous verrons la complexité de l'étude de ce modèle cousin du dilemme itéré du prisonnier via la sensibilité des résultats des simulations.

Ce modèle reste très intéressant, car en plus de fournir un second niveau de coopération, il permet en plus d'intégrer la communication entre agents. C'est sans doute ce qui en fait la différence la plus notable avec le modèle de base, mais aussi la plus intéressante.

Nous tirerons alors des conclusions des différents travaux exposés tout au long de la thèse en tentant d'y apporter des débuts de réponses aux problèmes de coopération entre agents. Nous insisterons sur la nécessité de diffusion de ces résultats, à travers notamment la définition d'un site web. Cette diffusion pouvant non seulement permettre la validation des résultats mais aussi l'utilisation de ces résultats dans des cas concrets de nécessité de coopération.

En annexe on trouvera la description des stratégies utilisables avec le simulateur ayant servi pour les travaux de cette thèse, ainsi que les algorithmes principaux qu'il utilise.

Chapitre 1

Théorie des jeux et automates

Dans ce premier chapitre, et après un court historique, nous allons rappeler brièvement quelques bases de mathématiques et d'informatique, concernant la théorie des jeux et les automates. Ces notions nous seront utiles tout au long de la thèse. Les notions de théorie des jeux introduites seront celles initiées par [vNM44] et raffinées dans [LR85].

1.1 Historique

L'approche mathématique moderne de l'étude des conflits d'intérêts, que l'on nomme théorie des jeux, est souvent attribuée à John von Neumann (1903-1957). Son article « *Zur Theorie der Gesellschaftspiele* » publié en 1928 est considéré comme étant à la base de toute cette théorie, même si les notions étudiées avaient déjà été présentées entre 1921 et 1927 par un mathématicien français, Émile Borel (1871-1956), dans une série d'articles sur *la théorie du jeu*.

C'est pourtant bien von Neumann qui, grâce à sa démonstration du théorème du *minimax* [2], a donné la respectabilité mathématique à cette théorie. De façon à pouvoir la répandre au-delà du monde fermé des mathématiciens, il a persévéré en signant avec l'économiste autrichien Oskar Morgenstern (1902-1976), le « *Theory of Games and Economic Behavior* » [vNM44], ouvrage qui aujourd'hui encore sert de base, sinon de référence.

Bien que les débuts de cet ouvrage aient été difficiles, du fait de la révolution dans les sciences économiques que les auteurs prétendaient apporter, la théorie des jeux est aujourd'hui très répandue et utilisée, non seulement en économie, mais également par toute une classe d'autres sciences dans lesquelles l'étude des situations de conflits est pertinente : sociologie, biologie, évolution, informatique.

En informatique, et plus précisément en Intelligence Artificielle (IA), les situations de conflits sont très fréquentes et l'apport de la théorie des jeux a été capital. On peut noter, par exemple, que pendant longtemps, les chercheurs ont essayé de construire des programmes capables de jouer aux échecs et, dans la mesure du possible, de *bien* jouer. Jusqu'à aujourd'hui, la majorité de ces algorithmes est basée sur l'application du théorème du minimax. Les bases d'un grand nombre d'algorithmes d'IA sont issues de la théorie des jeux.

Depuis la naissance de la discipline, et jusqu'à aujourd'hui, la théorie n'a cessé d'évoluer. Le but initial de ses fondateurs était de donner un sens à la notion de rationalité, notamment en ce qui concerne les interactions entre individus. C'est dans cet esprit que, dès le début, de nombreuses améliorations ont été apportées à la théorie des jeux. On peut notamment citer la *théorie de l'utilité* [3], ou la généralisation à des jeux à plus de deux joueurs. Au début des années 1980, après une période de stagnation, certains théoriciens des jeux, essentiellement des biologistes, ont décidé de changer la perspective de la théorie en s'intéressant à des jeux répétés, avec des comportements individuels prédéterminés relativement simples. En ramenant ainsi les stratégies à des automates, on ne s'intéresse plus aux choix possibles, mais aux résultats de ces choix. L'objectif de cette nouvelle approche est

2. Borel soupçonnait ce théorème être faux dans le cas général mais vrai dans certains cas particuliers
3. Elle a d'ailleurs été introduite dès 1947 dans la réédition de [vNM44].

d'étudier la stabilité des stratégies, ou plus globalement les équilibres du point de vue de l'évolution et non plus simplement du point de vue du jeu [4]. John Maynard Smith, biologiste anglais, a été un des fondateurs de cette nouvelle approche au travers des résultats publiés dans [Smi82].

Dans la mesure où les stratégies sont considérées comme étant des automates, et les jeux comme des répétitions de jeux simples, le recours aux simulations informatiques a été évident. Cette utilisation est d'autant plus justifiée que les résultats formels sur les dynamiques de population sont rares et peu informatifs.

1.2 Théorie des jeux

Dans l'introduction de [Gue95], on peut lire :

> Selon l'acceptation courante, un jeu est une situation où des individus (les « joueurs ») sont conduits à faire des choix parmi un certain nombre d'actions possibles, et dans un cadre défini à l'avance (les « règles du jeu »), le résultat de ces choix constituant une issue du jeu, à laquelle est associé un gain, positif ou négatif, pour chacun des participants.

Si l'on fait abstraction de l'idée de divertissement habituellement liée au mot, cette définition est mathématiquement satisfaisante. On peut cependant ajouter que les intérêts des *joueurs* sont souvent différents, et même opposés dans de nombreux cas, ce qui permet de considérer les jeux, ou en tout cas certains d'entre eux, comme les modèles les plus simples de situation de conflits d'intérêts.

1.2.1 Un premier exemple : Pierre/Ciseaux/Papier

Dans la suite de ce chapitre, nous allons introduire les définitions et notations de théorie des jeux que nous utiliserons dans la thèse. Pour illustrer ces définitions, voyons tout d'abord l'exemple d'un jeu particulier, simple et très connu : *Pierre, Ciseaux, Papier*.

Règles du jeu

Dans ce jeu, appelons-le PCP, deux joueurs, J_1 et J_2, s'affrontent. Ils doivent simultanément choisir un élément parmi les trois suivants :

1. Pierre
2. Ciseaux
3. Papier

S'ils choisissent le même élément, la partie est nulle, sinon on considère que :

- le papier enveloppe la pierre, et le joueur ayant choisi le papier gagne la partie ;
- les ciseaux découpent le papier, et le joueur ayant choisi les ciseaux gagne la partie ;
- la pierre détruit les ciseaux, et le joueur ayant choisi la pierre gagne la partie.

On dit que les actions disponibles pour chacun des joueurs sont des *stratégies*. Dans le jeu qui nous intéresse, chaque joueur peut donc jouer en choisissant une stratégies parmi les trois suivantes :

- prendre la pierre ;
- prendre les ciseaux ;
- prendre la feuille de papier.

Chaque joueur connaît les choix qu'il peut effectuer, les choix que son adversaire peut faire, ainsi que le résultat dans chacune des issues possibles : on dit alors que le jeu est à *information complète*.

L'intérêt de J_1 est complètement opposé à celui de J_2 : on dit que le jeu est *strictement compétitif*. Dans ce cas particulier la somme des utilités est toujours égale à zéro quel que soit l'issue : on dit que le jeu est à somme nulle.

4. Il est cependant évident que les deux notions sont liées.

Représentation extensive

On peut alors représenter les différentes parties possibles grâce à un arbre comme celui de la figure 1.1.

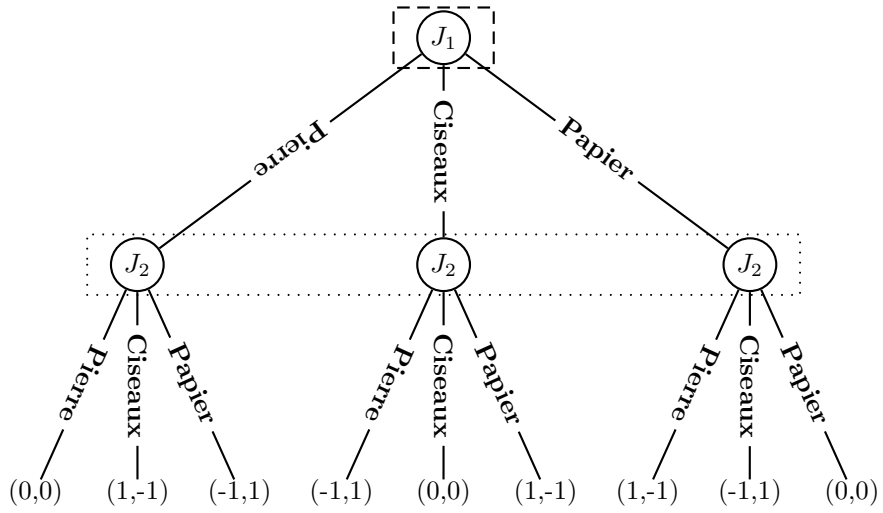


FIG. 1.1 – *Forme extensive du jeu Pierre/Ciseaux/Papier*

La racine de l'arbre correspond au coup du joueur J_1. Trois arcs, correspondant aux trois éléments que ce joueur peut choisir de prendre, en sont issus. Ces arcs rejoignent des nœuds, représentant les coups possibles du joueur J_2. De chacun de ces nœuds, trois nouveaux arcs rejoignent une feuille particulière de l'arbre, correspondant à un *résultat* du jeu, que l'on nomme communément une partie. Sur chaque arc est noté l'élément choisi par le joueur situé en amont de l'arc.

Il y a autant de parties possibles que de feuilles à l'arbre. À chaque partie est associée une *issue*, qui est en fait un vecteur des *utilités*, encore appelées *gains* ou *paiements*, de chacun des joueurs. Dans le cas de la figure 1.1, ces vecteurs donnent en première coordonnée le gain du joueur J_1, et en seconde coordonnées le gain du joueur J_2. Un gain de 1 signifie que le joueur a gagné, un gain de -1 signifie que le joueur a perdu, et un gain de 0 signifie que la partie est nulle.

Cette représentation est appelée la *forme extensive*, ou encore la *forme développée* du jeu PCP. L'arbre utilisé est appelé un *arbre de* KUHN [5].

Pour faire apparaître la simultanéité du jeu sur la représentation, on a entouré les *ensembles d'informations*. J_2 sait que J_1 a choisi un élément, mais il ne sait pas lequel, donc il ne connaît pas le nœud exact où son propre choix va intervenir, et donc il est incapable de déterminer l'issue du jeu qui va être atteinte. Le jeu est dit à *information imparfaite*.

Représentation stratégique

On peut également représenter ce même jeu par un tableau à double entrées, comme celui de le tableau 1.1.

		J_2		
		Pierre	**Ciseaux**	**Papier**
J_1	**Pierre**	(0,0)	(1,-1)	(-1,1)
	Ciseaux	(-1,1)	(0,0)	(1,-1)
	Papier	(1,-1)	(-1,1)	(0,0)

TAB. 1.1 – *Forme stratégique du jeu Pierre/Ciseaux/Papier*

5. Harold W. KUHN, mathématicien et économiste américain.

Chaque ligne du tableau représente une des stratégies que le joueur J_1 peut suivre, et chaque colonne représente une de celles que le joueur J_2 peut utiliser. Les cases du tableau correspondent alors à un résultat du jeu, et on y inscrit l'issue associée, avec la même convention sur l'ordre des coordonnées que précédemment.

Cette représentation, via une matrice de vecteur de paiements, est nommée la *forme stratégique*, ou encore la *forme normale* du jeu PCP. On dira également que la matrice représentée par le tableau 1.1 est la *matrice de gains* de PCP.

Le jeu que l'on vient de décrire est un jeu dont le déroulement ne comporte qu'une *étape* : le choix simultané d'un objet par les joueurs. Un jeu peut cependant comporter un nombre quelconque d'étapes, ce qui complexifie son étude du jeu, ne serait-ce que par le nombre de stratégies alors disponible.

1.2.2 Un second exemple : La guerre des sexes

Ce second exemple va nous permettre d'introduire les notions d'équilibres et de répétition dans les jeux.

Supposons qu'un homme (J_1) et une femme (J_2) aillent au cinéma. Une fois sur place, ils doivent choisir entre aller voir un documentaire ou une comédie. L'un des deux préfère les documentaires et l'autre les comédies, mais tous deux préfèrent voir un film ensemble que séparément.

Les stratégies disponibles pour chacun des deux joueurs, en considérant qu'ils font leur choix simultanément [6], sont alors :

- Aller voir un documentaire, on notera `Doc`.
- Aller voir une comédie, on notera `Com`.

On peut alors représenter cette situation de conflits d'intérêts par le jeu dont le tableau 1.2 représente la forme stratégique, et que nous appelerons GDS. Les gains de chaque joueur représentent le *niveau de plaisir* atteint par le joueur. Il est évident que les valeurs de ces gains sont moins importantes que l'ordre qu'elles impliquent sur la préférence que chaque joueur a pour chaque issue.

		J_2	
		`Doc`	`Com`
J_1	`Doc`	(2,3)	(1,1)
	`Com`	(1,1)	(3,2)

TAB. 1.2 – *Forme stratégique du jeu de la guerre des sexes*

Résolution du jeu

Avant tout, il est à noter que toutes les analyses faites en théorie des jeux considèrent que les joueurs sont *rationnels*, c'est-à-dire qu'entre deux issues possibles, ils préfèreront toujours celle qui maximise leur utilité. C'est une hypothèse très forte mais qu'il est difficile de relâcher si l'on veut pouvoir faire une analyse pertinente.

On peut remarquer que GDS n'est pas un jeu strictement compétitif, en effet les intérêts de J_1 et J_2 ne sont pas complètement opposés : les gains de J_1 et J_2 ne progressent pas dans des directions opposées.

Si l'on étudie le jeu par sa matrice de gains, on peut se rendre compte qu'il y a deux issues remarquables : (2,3) et (3,2). Ces deux issues sont intéressantes à double titre.

Tout d'abord, si on considère chacune de ces issues, on s'aperçoit rapidement qu'il est impossible, en supposant qu'une autre issue soit atteinte, d'améliorer le score de l'un des deux joueurs sans dimi-

6. Ce qui est peu vraissemblable dans un cas réel, la *galanterie* obligeant à désynchroniser le jeu au profit de la femme `:-)`

nuer le score de l'autre. On dit que ces deux issues sont PARETO-*optimales* ou PARETO-*efficientes* [7] . L'optimum de PARETO caractérise la rationalité collective.

De même, pour chacune de ces deux issues, les deux joueurs n'ont aucun regret quant à leur choix de stratégie. S'ils considèrent la stratégie de leur adversaire comme inéluctable, leur propre choix de stratégie est le meilleur possible. On dit que les deux issues sont des *équilibres de* NASH [8]. L'équilibre de NASH caractérise la rationalité individuelle.

Répétition du jeu

Comme PCP, GDS est un jeu à une seule étape. Supposons maintenant que le couple retourne au cinéma la semaine suivante, et qu'il doive à nouveau faire ce choix. On peut de nouveau représenter cette situation par un jeu, qui n'est en fait que la répétition de GDS, notons-le GDS^2.

GDS^2 a deux étapes. Si l'on considère que lors de la deuxième étape chacun des deux joueurs sait ce que l'autre a choisi lors de la première étape, les stratégies disponibles sont maintenant des stratégies conditionnelles : elles peuvent tenir compte des coups joués par l'adversaire lors des étapes précédentes.

La description de ces stratégies suit le schéma suivant : je joue X_1 au premier coup, puis si l'autre a choisi le documentaire lors de la première sortie, alors je joue X_{2d}, sinon je joue X_{2c}, avec X_1, X_{2d} et X_{2c} prenant leur valeur dans l'ensemble {`Doc`,`Com`}. On notera cette stratégie (X_1, X_{2d}|[X_1,`Doc`]; X_{2c}|[X_1,`Com`]). On pourra lire cette notation en faisant référence à une représentation sous forme d'arbre : « *je joue* X_1*, puis si nous nous retrouvons en* [X_1,`Doc`] *alors je joue* X_{2d}*, et si nous nous retrouvons en* [X_1,`Com`] *alors je joue* X_{2c} ». Dans le cas de GDS^2, on a donc 8 stratégies :

1. (`Doc`, `Doc`|[`Doc`,`Doc`]; `Doc`|[`Doc`,`Com`]) : je choisis toujours le documentaire
2. (`Doc`, `Doc`|[`Doc`,`Doc`]; `Com`|[`Doc`,`Com`]) : je choisis toujours le documentaire, sauf si la première fois je me suis retrouvé(e) seul(e)
3. (`Doc`, `Com`|[`Doc`,`Doc`]; `Doc`|[`Doc`,`Com`]) : je choisis toujours le documentaire, sauf si la première fois nous avons tous les deux choisi le documentaire
4. (`Doc`, `Com`|[`Doc`,`Doc`]; `Com`|[`Doc`,`Com`]) : la première fois je choisis le documentaire et la seconde la comédie
5. (`Com`, `Doc`|[`Com`,`Doc`]; `Doc`|[`Com`,`Com`]) : la première fois je choisis la comédie et la seconde le documentaire
6. (`Com`, `Doc`|[`Com`,`Doc`]; `Com`|[`Com`,`Com`]) : je choisis toujours la comédie, sauf si la première fois je me suis retrouvé(e) seul(e)
7. (`Com`, `Com`|[`Com`,`Doc`]; `Doc`|[`Com`,`Com`]) : je choisis toujours la comédie sauf si la première fois nous avons tous les deux choisi la comédie
8. (`Com`, `Com`|[`Com`,`Doc`]; `Com`|[`Com`,`Com`]) : je choisis toujours la comédie

Pour chaque issue de GDS^2, les vecteurs d'utilité sont déterminés en effectuant la somme des vecteurs obtenus pour chacune des étapes considérées comme des issues de GDS. On dira que GDS^2 est un *superjeu* dont GDS est le *jeu constitutif*.

Dans le cas de jeux répétés, la notion d'équilibre de NASH perd un peu de sa force. En effet deux stratégies sont en équilibre de NASH si chaque joueur considère que sa stratégie est la meilleure réponse possible à la stratégie de l'adversaire. Dans un jeu répété, plusieurs combinaisons stratégiques peuvent aboutir à un même déroulement de jeu. On peut en observer un exemple en considérant les deux combinaisons stratégiques pour GDS^2 (1,1) et (1,2). Le déroulement du jeu dans les deux cas correspond à aller voir un documentaire dans les deux sorties. On a donc un déroulement de jeu identique pour deux combinaisons stratégiques différentes. Rien n'empêcherait donc d'avoir un grand nombre d'équilibre de NASH dans un tel jeu, et pourtant les stratégies utilisées dans chacune des

7. Vilfredo PARETO (1848-1923) économiste et sociologue italien
8. John NASH (1928-) économiste américain

combinaisons stratégiques ne peuvent pas être considérées comme autant *rationelles* les unes que les autres.

Dans les jeux répétés, on remplacera donc souvent la recherche d'équilibre de NASH par la recherche d'*équilibre parfait en sous-jeux*. Ces équilibres sont facilement définissables avec les représentations de jeux sous forme d'arbres de KUHN. Un équilibre parfait en sous-jeux correspond alors simplement à une combinaison stratégique dont les actions choisies pour chaque sous-jeux sont des équilibres de NASH. Un *sous-jeu* est simplement un sous-arbre de l'arbre de jeu.

1.2.3 Définitions

Après avoir vu deux exemples concrets présentant les notions de base de la théorie des jeux, nous en détaillons les concepts sous un aspect plus formel.

Description

Nous avons donc un certain nombre d'éléments qui composent un jeu :
- les joueurs ;
- les actions et stratégies des joueurs ;
- le déroulement et les étapes du jeu ;
- les résultats du jeu ;
- les informations dont disposent les joueurs lors de chaque choix d'action.

On peut alors donner les quelques définitions qui suivent.

Définition 1.1 *Les* **règles** *d'un jeu indiquent :*
1. *la succession des étapes du jeu, et l'ordre dans lequel interviennent les joueurs ;*
2. *les actions qui sont autorisées à chaque étape ;*
3. *les informations dont dispose le joueur chaque fois qu'il doit prendre une décision.*

Nous avons vu qu'il y a deux formes de représentations possibles pour un jeu. L'une d'entre elles utilise un arbre et l'autre une matrice.

Définition 1.2 *Un* **arbre de jeu** $A = (D,I,p)$ *est la donnée :*
- *d'un ensemble* D *de* nœuds de décisions*, ou* situations de jeu *;*
- *d'un ensemble* I *d'*issues *de jeu, avec* $D \cap I = \emptyset$ *;*
- *d'un élément* d_0 *de* D *et d'une fonction* p *de* $D \cup I - d_0$ *dans* D*, appelée fonction prédécesseur, qui pour chaque situation de jeu, ou issue, indique l'unique action qui permet d'arriver à cette situation, ou issue.*

Pour déterminer l'issue d'un jeu, il suffit de connaître les stratégies utilisées par chacun des joueurs. Une stratégie est une combinaison d'actions autorisées par les règles du jeu jusqu'à la fin de celui-ci. Il existe plus précisément trois types de stratégies.

Définition 1.3 *Une* **stratégie pure** s *d'un joueur* n *est une application de l'ensemble* D_n *des nœuds de décision du joueur* n *vers l'ensemble* D *de tous les nœuds de décision du jeu telle que :*

$$\forall d \in D_n,\ p(s(d)) = d$$

avec D_n *l'ensemble des nœuds de décision où c'est au joueur* n *de décider, et* p *la fonction prédécesseur du jeu.*

Plus simplement une stratégie pure est une stratégie ne faisant intervenir aucune forme de hasard, qui est donc complétement déterministe.

Définition 1.4 *Une* **stratégie mixte** *pour un joueur* n *est une distribution de probabilité sur l'ensemble de ses stratégies pures.*

Une stratégie mixte est donc une stratégie faisant le choix d'une parmi toutes les stratégies pures et qui utilise celle-ci durant toute la durée de jeu. Un joueur utilisant une stratégie mixte face à un joueur utilisant une stratégie pure utilisera donc lui aussi toujours une stratégie pure pour une rencontre, mais n'utilisera pas toujours la même stratégie pure lors de toutes leurs rencontres.

Définition 1.5 *Une* **stratégie de comportement** *pour un joueur n est un ensemble $C_n = \{\ldots,c_i,\ldots\}$ où i est un élément de D_n et c_i une distribution de probabilité sur le sous-ensemble $p^{-1}(i)$ des successeurs du nœud de décision i.*

Une stratégie de comportement est en fait le remplacement des choix certains d'une stratégie pure par des choix aléatoires. Dans la plupart des jeux qui seront étudiés dans la thèse, il existera une bijection entre l'ensemble des stratégies mixtes et l'ensemble des stratégies de comportement, on mélangera donc les deux termes en retenant celui de stratégies mixtes comme représentant toutes les stratégies faisant intervenir le hasard.

On dira qu'une ***combinaison stratégique*** est un vecteur de stratégies dont chaque élément correspond à la stratégie utilisée par un joueur participant au jeu. La donnée d'une combinaison stratégique détermine donc de manière complète l'issue d'un jeu. Si on se limite aux stratégies pures, on peut même dire que l'ensemble des combinaisons stratégiques est l'ensemble des issues. Par souci de simplification, on identifiera les combinaisons stratégiques à l'issue qu'elles déterminent, et plus généralement l'ensemble des combinaisons stratégiques à l'ensemble des issues d'un jeu.

Les joueurs doivent avoir des préférences parmi les issues qui sont à leur portée. C'est avec la définition de ces préférences que l'on peut caractériser la rationalité d'un joueur. La relation de préférence, que nous noterons $\succeq$, est une relation binaire sur l'ensemble des issues d'un jeu. On notera $x \succeq y$ et on lira « *x est au moins aussi bon que y* ». On peut alors définir la préférence stricte par la relation $\succ$ telle que :

$$x \succ y \iff x \succeq y \text{ mais pas } y \succeq x$$

que l'on lira « *x est préféré à y* », et la relation d'indifférence :

$$x \sim y \iff x \succeq y \text{ et } y \succeq x$$

Définition 1.6 *Une relation de préférence $\succeq$ est dite* **rationelle** *si elle est complète et transitive.*

Définition 1.7 *Une* **fonction d'utilité**, *ou encore* **fonction de paiement** [9] *est une fonction de l'ensemble des issues d'un jeu à n joueurs vers $\mathbb{R}^n$ qui associe les utilités retirées par chaque joueur de la réalisation de chaque issue.*

Si u est une fonction d'utilité on notera u_n la fonction de l'ensemble des issues d'un jeu vers $\mathbb{R}$ correspondant aux utilités du joueur n. Une telle fonction sera dite représentant de la relation de préférence $\succeq$ si pour toute issue x et y on a :

$$x \succeq y \iff u_n(x) \geq u_n(y)$$

La théorie de l'utilité qu'utilise la théorie des jeux axiomatise le fait que seule cette notion de préférence est importante. En bref, on peut dire que seul l'ordre de préférence des issues est important, la valeur des gains apportés par chaque issue est sans importance.

On peut donc désormais définir les deux approches d'un jeu.

Définition 1.8 *Un* **jeu sous forme développée** *$J = (A,N,u,F)$ est la donnée :*

- *d'un arbre de jeu $A = (D,I,p)$;*
- *d'un ensemble N de joueurs ;*
- *d'une fonction d'utilité u ;*

9. *payoff function* en anglais.

– *d'un ensemble de partition d'informations F, dont chaque élément est une partition de D et indique les états du jeu que le joueur est capable de distinguer.*

Un jeu sous forme développée est également dit sous forme extensive, ou sous forme d'arbre de KUHN.

Un jeu est à ***information complète*** quand chaque joueur connaît l'ensemble des composantes du jeu, et à ***information incomplète*** sinon. Il est à noter que dans le cas d'information complète F ne contient qu'une seule partition, ce qui revient à dire que les joueurs n'ont qu'une seule vue sur l'arbre de jeu.

Un jeu est à ***information parfaite*** quand l'unique élément de F se réduit à une partition de D où chaque nœud de décision forme un sous-ensemble, c'est à dire que chaque élément de la partition est un nœud de l'arbre et réciproquement. Plus simplement on peut dire que dans ce cas les joueurs peuvent savoir à chaque instant quel nœud de l'arbre est atteint. Dans le cas contraire le jeu est dit à ***information imparfaite***.

On peut remarquer que tous les jeux simultanés, c'est-à-dire dans lesquels les joueurs font leur choix en même temps, sont des jeux à information imparfaite. En effet au moment de son choix, le joueur ne sait pas sur quel nœud de décision il se trouve.

Définition 1.9 *Un* **jeu sous forme normale** $J = (N,S,u)$ *est la donnée :*

– *d'un ensemble N de joueurs ;*
– *d'un ensemble S de combinaisons stratégiques ;*
– *d'une fonction d'utilité u définie sur S.*

Un jeu sous forme normale est également dit sous forme stratégique. On simplifie d'ailleurs la donnée du jeu à la donnée de la fonction d'utilité, sous la forme d'une matrice de paiement.

Dans la suite de la thèse, nous nous intéresserons essentiellement aux jeux à deux joueurs, dont les stratégies accessibles pour les deux joueurs sont identiques. Un certain nombre de simplification sera donc possible. Pour un tel jeu, si `A` et `B` sont deux stratégies on notera V(`A`|`B`) l'utilité du joueur ayant choisi la stratégie `A` dans l'issue associée à la combinaison stratégique (`A`,`B`). On lira score de `A` face à `B`.

Définition 1.10 *Un jeu est* **concurrentiel pur**, *ou* **strictement compétitif** *si :*

$$\forall (i,j) \in I \times I,\ \exists (m,n) \in N \times N,\ (u_m(j) - u_m(i))\,(u_n(j) - u_n(i)) < 0$$

avec N l'ensemble des joueurs.

En fait, un jeu est concurrentiel pur si les utilités des joueurs progressent dans des directions opposées. Parmi les jeux concurrentiels purs, on peut distinguer une classe particulière de jeu : les ***jeux à somme nulle***. Un jeu est à somme nulle si

$$\forall i \in I,\ \sum_{n \in N} u_n(i) = 0$$

Un jeu répété est la répétition d'un jeu un certain nombre de fois. La fonction d'utilité associée aux issues de la répétition du jeu dépend directement de la fonction d'utilité du jeu de base, que l'on nommera ***jeu constitutif***. C'est en fait une somme pondérée des utilités obtenues pour chaque issue du jeu constitutif, c'est-à-dire de chaque coup. Le jeu répété sera nommé ***superjeu***. Plus précisément :

Définition 1.11 *Un* **superjeu** $\mathcal{J} = (J,T,\Omega)$ *est la donnée :*

– *d'un jeu constitutif $J = (N,S,u)$;*
– *du nombre de répétitions T ;*
– *du vecteur $\Omega = (\omega_1, \ldots, \omega_n)$ de taux d'escompte d'utilité, ω_i étant le taux d'escompte du joueur i.*

Si on considère qu'à une étape t du superjeu $\mathcal{J}$ le choix dicté par une combinaison stratégique s au joueur n est noté $s_{n,t}$ et que l'utilité, pour ce même joueur, obtenue à cette étape du jeu, c'est-à-dire l'utilité de l'issue du jeu constitutif correspondant, est notée $u_n(s_{n,t})$, alors l'utilité associée à l'issue du superjeu est :

$$u_n = \sum_{t=0}^{t=T-1} \omega_n^t u_n(s_{n,t})$$

Résolution

La résolution d'un jeu correspond à la recherche d'une issue et d'une combinaison stratégique qui permettent aux joueurs de maximiser leur utilité, ce qui correspond à une certain forme de rationalité pour les joueurs. On considèrera donc qu'être rationel pour un joueur, c'est le fait de préférer entre deux issues celle qui offre la plus grande utilité.

Cette rationalité peut se voir appliquer à au moins deux niveaux :

– au niveau collectif ;

– au niveau individuel.

Dans le premier cas on appliquera le principe de rationalité à la collectivité des joueurs et on aboutira alors à l'optimalité selon Pareto.

Dans le cas de l'application du principe de rationalité au joueur en tant qu'individu, et non plus membre d'une collectivité, on aboutira à la définition d'un *équilibre de* Nash.

Définition 1.12 *Une issue i d'un jeu J est* **pareto-optimale** *si*

$$\forall j \in I - \{i\},\ \exists n \in N,\ u_n(j) < u_n(i)$$

avec I l'ensemble des issues de J, N l'ensemble des joueurs et u la fonction d'utilité.

Ce qui revient à dire qu'une issue n'est pas Pareto-optimale s'il existe au moins une issue préférée par l'ensemble des joueurs. On peut remarquer que toutes les issues d'un jeu strictement compétitifs sont Pareto-optimales, qu'un jeu peut n'en avoir aucune, une ou plusieurs.

La Pareto-optimalité caractérise une issue mais ne dit rien sur la méthode à utiliser par les joueurs pour l'atteindre de manière certaine, en tout cas dans les jeux que nous étudierons, à savoir quand les joueurs n'ont aucun moyen de communication, sinon l'historique des coups joués dans le passé. On appellera ces jeux des jeux *non-coopératifs*.

Définition 1.13 *Une stratégie s domine une stratégie t pour un joueur n si et seulement si :*

$$\forall s_n^* \in \prod_{i \in I}^{i \neq n} S_i,\ u_n(s,s_n^*) \geq u_n(t,s_n^*)$$

et

$$\exists t_n^* \in \prod_{i \in I}^{i \neq n} S_i,\ u_n(s,t_n^*) > u_n(t,t_n^*)$$

avec S_i l'ensemble des stratégies pour le joueur i, s_i^* un vecteur de stratégies pour tous les joueurs hormis le joueur i, et $u_i(s,s_i^*)$ l'utilité pour le joueur i de l'issue déterminée par la combinaison stratégique définie par la stratégie s pour le joueur i, et le vecteur de stratégies s_i^* pour les autres joueurs.

Plus simplement une stratégie s domine une stratégie t pour le joueur n si toutes les combinaisons stratégiques faisant intervenir s pour n rapportent une utilité plus importante ou égale que celles faisant intervenir t, et qu'au moins une des issues faisant intervenir s rapporte strictement plus qu'une autre faisant intervenir t.

Si toutes les inégalités sont strictes on dira que s domine strictement t.

Une des méthodes de résolution d'un jeu est d'éliminer les stratégies dominées itérativement pour chaque joueur. En étudiant la représentation stratégique d'un jeu un joueur peut, par exemple, réduire l'étude de l'espace de choix de son adversaire en éliminant les stratégies dominées. Cette réduction peut alors lui faire apparaître des dominations dans son espace de stratégies, et lui permettre de choisir la meilleure stratégie à utiliser. Son adversaire peut en faire de même. Le choix de chacun des deux joueurs après cette étude sera alors considéré comme une solution dite par *élimination successive des stratégies dominées*.

Un joueur privilégiant la rationalité individuelle ne jouera donc jamais une stratégie dominée. Pour chaque combinaison stratégique des autres joueurs, ce que l'on a noté s_n^*, un joueur est capable de déterminer sa meilleure réponse, c'est-à-dire la stratégie qui maximisera son utilité dans l'issue déterminée par cette combinaison stratégique. Dans une représentation sous forme stratégique cela revient simplement à choisir pour une colonne donnée la ligne qui rapporte l'utilité la plus grande.

Définition 1.14 *Un* **équilibre de Nash** *est une combinaison stratégique où chaque joueur joue sa meilleure réponse contre tous les autres.*

On peut exprimer la définition d'un équilibre de Nash, comme étant une *situation de non regret* : chaque joueur ne regrette pas son choix stratégique au vu du choix des autres joueurs.

Tous les jeux ne possèdent pas d'équilibre de Nash et certains jeux en possèdent plusieurs. Tous les équilibres de Nash ne sont pas optimaux au sens de Pareto comme nous le verrons avec le dilemme du prisonnier.

L'équilibre de Nash est un concept central en théorie des jeux. C'est sans doute **le** concept de solution que préconise la théorie si on choisit la rationalité individuelle en dehors de toute autre considération philosophique.

Définition 1.15 *Un* **équilibre parfait en sous-jeux** *est une combinaison stratégique où à chaque étape d'un jeu répété un équilibre de* Nash *est atteint.*

La recherche de tels équilibres n'est pas toujours simple, notamment pour les jeux à information imparfaite.

Nous arrêtons là les rappels de théorie des jeux qui sont suffisants pour comprendre les jeux que nous étudierons par la suite. Les propriétés et autres théorèmes utilisés seront introduits au moment de leur utilisation.

Pour un approfondissement des idées et résultats de la théorie des jeux, on pourra se diriger vers les références suivantes :

- [vNM44] est l'ouvrage fondateur de la théorie des jeux, mais aussi celui qui tourna résolument la théorie vers le monde des économistes, de la volonté même de ses auteurs. Malheureusement cet ouvrage est difficilement accessible et semble de toute façon relativement incomplet puisque de nombreuses avancées dans la théorie, dont la notion d'équilibre de Nash sont apparues après sa publication. Je n'ai jamais eu la possibilité de le consulter.
- [LR85] est une introduction particulièrement didactique et pédagogique de la théorie des jeux comme étant la science du choix sous contraintes. L'ouvrage est un des plus anciens recueils d'informations sur la théorie des jeux. Il est fort peu formalisé, son but avoué étant de faire comprendre les concepts plutôt que de les décrire précisément, les annexes couvrant plus du tiers du livre se chargeant de ces détails. Un survol complet de la théorie de jeux y est fait, de la théorie de l'utilité, jusqu'aux applications de la théorie des jeux impliquant plus de deux joueurs. En plus de sa conception pédagogique, le second avantage de ce livre est constitué par les critiques que les auteurs font des choix qui sont faits en théorie des jeux, notamment en ce qui concerne les concepts de rationalités. Certaines tentatives de réponses à ces critiques sont d'ailleurs apportées, même si la plupart du temps la critique est plus développée que l'éventuelle correction.

– [Gue95] est un très court ouvrage de présentation rapide de la théorie des jeux et de ses principaux concepts. Loin d'être une référence exhaustive il est cependant clair, concis et à peu près complet. Les concepts y sont toujours expliqués sur la base d'exemples et avec fort peu de formalisme mathématique. L'ouvrage est divisé en quatre chapitres qui présentent chacun une grande classe de jeux. Pour chacun d'entre eux après une présentation du type de jeu, l'auteur explique ce que la théorie des jeux a réussi à apporter, mais aussi où son apport s'arrête. Enfin un point sur l'état actuel de la théorie des jeux est fait, et notamment sur les jeux répétés qui sont au centre de la discussion de cette thèse.
– [Bin99] est un volumineux ouvrage qui se veut être un cours complet de présentation des jeux et de la théorie des jeux. Il couvre un spectre très large de catégories de jeux, avec peut-être certaines simplifications ou absences sur les apports des aspects expérimentaux de la théorie des jeux. De plus dans le cadre qui nous intéresse, à savoir les jeux répétés, les stratégies décrites portent parfois des noms peu usités dans la littérature, obligeant le lecteur à une gymnastique fort peu agréable. Le point fort de cet ouvrage reste néanmoins les descriptions mathématiques formelles des diverses notions de théorie des jeux. L'objectif de l'ouvrage étant d'être un support de cours de théorie des jeux destiné aux étudiants non mathématiciens, de nombreux rappels sont faits avant l'utilisation d'outils mathématiques. On peut cependant regretter le ton volontairement plus *oral* que *littéraire* du texte. Les notations et formalisations utilisées sont en accord avec [MCWG95], ouvrage de référence en théorie micro-économique, et on peut donc dire que ce sont celles adoptées par les économistes.
– [Jay96] est un support de cours de théorie des jeux destiné à des économistes. Les deux premiers chapitres présentent successivement les jeux, et les principes de résolution des jeux. Cette présentation est faite non seulement via des exemples, mais aussi par des tentatives de formalisation mathématique. Les chapitres suivants sont plus destinés à montrer l'intérêt de la théorie des jeux au sein des sciences économiques et sont donc moins intéressants du point de vue de notre discussion.

Il reste à noter que la quasi-totalité de ces ouvrages s'adressent avant tout aux économistes. Le vocabulaire, tout comme la nature de certains jeux pris en exemple, peuvent dérouter les non-économistes au premier abord.

1.3 Automates

Dans de nombreux cas, et notamment dans le cas des jeux répétés, les stratégies disponibles pour les joueurs sont considérées comme étant des automates finis. Les automates utilisés sont des *automates d'états finis* (*finite state automata* ou *finite state machine* en anglais) avec sortie. Nous utiliserons deux types d'automates finis, à savoir les automates de Moore et les automates de Mealy.

Définition 1.16 *Un* **automate fini avec sorties** $M = (\mathcal{Q},q_0,\Sigma,\Delta,\lambda,\delta)$ *est la donnée :*
– *d'un ensemble non vide et fini* $\mathcal{Q}$ *d'états ;*
– *d'un* état initial q_0 *de* $\mathcal{Q}$ *dans lequel se trouve l'automate au moment de son démarrage ;*
– *d'un ensemble fini de symboles d'entrées* Σ *;*
– *d'un ensemble fini de symboles de sorties* Δ *;*
– *d'une* fonction de sortie λ *permettant de connaître la valeur de sortie de l'automate ;*
– *d'une* fonction de transition $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ *permettant de calculer le nouvel état de l'automate en fonction des entrées.*

Σ et Δ sont appelés des alphabets.

Définition 1.17 *Un* **automate de Moore** $M = (\mathcal{Q},q_0,\Sigma,\Delta,\lambda_{\text{mo}},\delta)$ *est un automate fini avec sorties dont la fonction de sortie est :*

$$\lambda_{\text{mo}} : \mathcal{Q} \to \Delta$$

Dans un automate de MOORE la sortie de l'automate dépend donc uniquement de l'état dans lequel se trouve l'automate.

Définition 1.18 *Un* **automate de Mealy** $M = (\mathcal{Q},q_0,\Sigma,\Delta,\lambda_{\text{me}},\delta)$ *est un automate fini avec sorties dont la fonction de sortie est :*

$$\lambda_{\text{me}} : \mathcal{Q} \times \Sigma \to \Delta$$

Dans un automate de MEALY la sortie de l'automate dépend non seulement de l'état dans lequel se trouve l'automate, mais aussi des entrées de l'automate.

On préférera souvent représenter les automates au moyen d'un graphe. Chaque état de l'automate est représenté par un sommet du graphe. La fonction de transition est représentée par des arcs étiquetés. Par exemple si $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ et que l'on a $\delta(p,\sigma) = q$, alors le graphe comporte un arc étiqueté par σ reliant les sommets p et q. Par convention l'état initial est représenté par une flèche double. Pour les automates de MOORE on associera souvent le nom de l'état à la sortie qui lui est associée par la fonction λ. Pour les automates de MEALY on associera souvent les fonctions de sortie et de transition, en étiquetant les arcs avec non seulement les entrées mais aussi les sorties.

Pratiquement, on peut voir sur la figure 1.2 un exemple d'automate de *Moore* représentant une stratégie pour le jeu GDS^2. Il s'agit plus exactement de la stratégie 2.

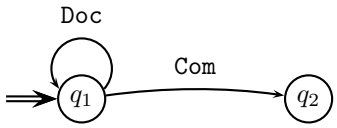


FIG. 1.2 – *Une stratégie pour GDS^2 sous la forme d'un automate d'état de* MOORE

On peut voir que $\Sigma = \Delta = \{\texttt{Doc},\texttt{Com}\}$, que $\mathcal{Q} = \{q_1,q_2\}$ et que l'état initial est l'état q_1. La fonction de sortie est simplement :

$$\begin{aligned} \lambda \;:\; & \mathcal{Q} \to \Delta \\ & q_1 \mapsto \texttt{Doc} \\ & q_2 \mapsto \texttt{Com} \end{aligned}$$

La fonction de transition, quant à elle, est simplement :

$$\begin{aligned} \delta \;:\; & \mathcal{Q} \times \Sigma \to \Delta \\ & (q_1,\texttt{Doc}) \mapsto q_1 \\ & (q_1,\texttt{Com}) \mapsto q_2 \end{aligned}$$

On peut voir sur la figure 1.3 La représentation de la même stratégie sous la forme d'un automate de MEALY.

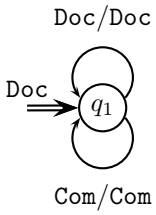


FIG. 1.3 – *Une stratégie pour GDS^2 sous la forme d'un automate d'état de* MEALY

On peut voir sur cet exemple, que $\Sigma = \Delta = \{\texttt{Doc},\texttt{Com}\}$, que $\mathcal{Q} = \{q_1\}$ et que l'état initial est l'état q_1. La fonction de sortie est simplement :

$$\begin{aligned} \lambda \quad : \quad & \mathcal{Q} \rightarrow \Delta \\ & (q_1,\texttt{Doc}) \mapsto \texttt{Doc} \\ & (q_1,\texttt{Com}) \mapsto \texttt{Com} \end{aligned}$$

La fonction de transition, quant à elle, est simplement :

$$\begin{aligned} \delta \quad : \quad & \mathcal{Q} \times \Sigma \rightarrow \Delta \\ & (q_1,\texttt{Doc}) \mapsto q_1 \\ & (q_1,\texttt{Com}) \mapsto q_1 \end{aligned}$$

Une petite adaptation est faite, à savoir que l'état d'entrée de l'automate est atteint en émettant une sortie, de façon à pouvoir initialiser l'automate.

Le fait que le nombre d'états soit fini est une contrainte forte. Il implique que l'on peut déterminer complètement la séquence des états parcourus pour des entrées données. Cela implique aussi que toute séquence infinie est en fait composée de répétitions. En effet si une séquence est infinie, comme le nombre d'états est fini, un même état sera forcément atteint deux fois. La séquence de transitions utilisée entre ces deux étapes se répètera donc indéfiniment.

C'est avec le rappel de cette propriété importante que nous terminerons ce chapitre. Après avoir fait ces rappels, nous allons pouvoir présenter le jeu qui sert de modèle à l'étude de la coopération entre agents et qui constitue le cœur de la thèse.

Chapitre 2

Dilemme du prisonnier

Nous présentons maintenant le modèle d'étude de la coopération sur lequel se base la thèse. Nous faisons pour cela un rappel historique sur l'apparition du modèle, présenté pour la première fois dans [Flo52], qui permettra d'en comprendre les fondements et objectifs ; puis nous survolons quelques idées rencontrées dans la littérature à son propos ; et enfin nous apportons certaines précisions originales sur le modèle.

2.1 Apparition du dilemme

2.1.1 La RAND Corporation

Peu après la seconde guerre mondiale, l'armée américaine, (plus exactement l'US Air Force), a soutenu la création d'un organisme dont le but original était *l'étude des stratégies de guerre nucléaire intercontinentale*. Cet organisme, créé officiellement en 1948 et qui existe toujours aujourd'hui, mais dont les objectifs ont quelque peu évolué, se nomme la *RAND Corporation* [10]. À l'origine, RAND est une sorte d'acronyme, comme les apprécient les américains, contraction de *Research ANd Development*. Cet organisme est en fait une entreprise privée dont la majorité des fonds est venue, au début, de la compagnie aéronautique et d'armement Douglas Aircraft. La méthode de fonctionnement de cette société était d'engager à plein temps des chercheurs qui quittaient les programmes de recherche militaire (comme le projet Manhatan à l'origine des premières bombes A et H), mais aussi et surtout d'engager comme consultants des chercheurs universitaires, notamment pendant les périodes estivales. C'est ainsi qu'un grand nombre de chercheurs se retrouvèrent dans les locaux de la RAND Corp à Santa-Monica, en Californie, durant les étés de la fin des années cinquante et du début des années soixante.

Les responsables de la RAND Corp étaient suffisamment visionnaires pour ne pas se focaliser sur la recherche directement appliquée aux techniques militaires. De nombreux programmes de recherche fondamentale y ont vu le jour. L'intérêt de la théorie des jeux, comme une formalisation possible de la modélisation des situations de conflits d'intérêts, y a notamment été relevé très tôt. Dès le début, on a pu compter parmi les chercheurs consultants très actifs des personnalités comme von Neumann, Morgenstern ou encore Nash. On peut trouver plus de précisions sur la constitution de la RAND Corp dans le chapitre 5 de [Pou92] qui est un ouvrage à la fois biographique de John von Neumann et une présentation du dilemme du prisonnier, le tout d'une manière très agréable, proche du roman.

La position de certains chercheurs vis-à-vis de la théorie des jeux a cependant été très critique. Les critiques sont plus basées sur son applicabilité que sur sa rigueur mathématique irréprochable. Le principal reproche fait à la théorie est son manque d'apport en terme de prévision de jeu pour des joueurs humains. On peut par exemple lire dans l'introduction de [Flo52] :

> « *The two-person zero-sum case seems to be useful in understanding certain parlor games. Even here there is room for doubt since the theory neither predicts the outcome of*

10. `http://www.rand.org`

a chosen method of play, nor describes how persons will actually play, in even the simplest games. »

Il est vrai que la théorie ne dit jamais comment un joueur va jouer, mais plutôt comment il devrait jouer dans des circonstances particulières. En effet, la théorie se base sur le concept de rationalité des joueurs participants. Or, pour ces chercheurs, il semblait évident que les humains ne sont pas rationnels, et donc que la théorie de départ n'apportait pas grand chose. De plus, cette notion de rationalité est basée sur la notion d'utilité, qui selon ces critiques est en elle-même souvent inapplicable. Toujours dans l'introduction de [Flo52] on peut lire à ce propos :

« *The utility concept, as it enters into game theory, can be criticized on the basis that suitable operational measures often cannot be found in real life applications. Indeed, in mosts cases, it is far more difficult to construct an acceptable accounting procedure for recording utilities in the attempted application that it would be to find a reasonably good strategy in the absence of game theory.* »

Le système d'axiomes de la théorie des jeux impose un cadre d'étude et des contraintes qui semblaient trop fortes pour l'objectif que VON NEUMANN et MORGENSTERN se sont donnés lors de la rédaction de [vNM44], à savoir l'étude des comportements économiques humains.

2.1.2 Les expériences de Flood et Dresher

Afin de tenter d'étayer ces critiques et d'apporter des arguments convaincants, Merill FLOOD a essayé de tester la théorie dans des conditions expérimentales concrètes. Une compilation de ses expériences a été publiée en 1952, dans un mémoire de recherche de la RAND Corp, [Flo52]. Dans ce mémoire FLOOD décrit précisément 7 expériences différentes. Le but de chacune d'elles était non seulement de tester la théorie des jeux en vraie grandeur, et donc en utilisant des humains, mais aussi et surtout d'essayer de trouver les modifications à y apporter afin de pouvoir y introduire un peu d'irrationalité, ou en tout cas de *non-rationalité*, pour pouvoir mieux étudier les comportements humains.

L'illustration la plus flagrante, et en tout cas la plus simple, de la non adéquation de la théorie à la pratique est donnée par le test numéro 4 de [Flo52]. La description qui en est faite est la suivante :

« *The experimenter E offers to give Subject 1 an amount m but to give Subjects 1 and 2 together a greater amount $m + g$ if they can agree on sharing the larger amount.* »

Le jeu utilisé dans cette expérience implique donc deux joueurs, soit A et B. L'expérimentateur doit offrir de donner une certaine somme s_A à A en précisant que si A et B arrivent à s'entendre sur un partage alors la somme distribuée globalement aux deux sera $S > s_A$. Une très jolie formalisation de ce jeu est donnée dans [Flo52]. FLOOD a essayé cette expérience deux fois sur deux secrétaires partageant le même bureau à la RAND Corp. Dans une des expériences, $s_A = \$0.5$ et $S = \$1.5$. Si l'on respecte complètement la théorie des jeux, à savoir le principe de rationalité des joueurs, si le partage se fait, alors le joueur A doit recevoir \$1, c'est-à-dire son gain minimal additionné de la moitié du gain offert en plus pour le partage. En utilisant la formalisation de FLOOD ce couple de stratégies est un équilibre de NASH. Le joueur B doit obtenir \$0.5, c'est-à-dire la moitié du gain supplémentaire. En utilisant la formalisation de FLOOD ce couple de stratégies est un équilibre de NASH. Lors de l'expérience, les deux secrétaires se sont entendues sur un partage complètement équitable, c'est-à-dire \$0.75 chacune. On voit bien sur ce petit exemple que la théorie n'arrive pas à atteindre son objectif.

Le test numéro 5 de [Flo52], effectué en janvier 1950, en collaboration avec Melvin DRESHER, décrit un jeu dans un cadre beaucoup plus fermé que le précédent. C'est un jeu :

– **répété**, le jeu constitutif est répété un nombre fini de fois, dans les expériences de [Flo52] la répétition est de 100 fois ;

- **non coopératif**, les joueurs ne peuvent pas communiquer, et ne peuvent donc pas s'entendre sur la stratégie à adopter ;
- **simultané**, les joueurs font leur choix en même temps en ne connaissant rien du choix de leur adversaire ;
- **à somme non nulle**, les intérêts des joueurs ne sont pas diamétralement opposés. Le jeu n'est pas concurrentiel pur.

L'expérience a été faite sur deux chercheurs alors présents à la RAND Corp : Armen Alchian, de l'université de Californie à Los Angeles, noté AA dans les expériences, et John D. Williams, de la RAND Corp, noté JW. La fonction d'utilité du jeu constitutif utilisée est décrite par le tableau 2.1. Cette fonction d'utilité donnée sous la forme de la matrice des gains était connue de chacun des deux *cobayes*. Elle est volontairement un peu complexe, à commencer par sa non-symétrie dans les gains, afin de rendre l'analyse un peu plus complexe pour les deux joueurs, qui ne connaissaient pas encore le concept d'équilibre de NASH, et donc laisser leur *irrationalité humaine* intervenir un peu dans leur choix.

	$\mathbf{JW}_1$	$\mathbf{JW}_2$
$\mathbf{AA}_1$	(-1,2)	$(\frac{1}{2},1)$
$\mathbf{AA}_2$	$(0,\frac{1}{2})$	(1,-1)

TAB. 2.1 – *Fonction d'utilité du jeu de l'expérience 5 de [Flo52]. Les gains de AA sont données en premier.*

L'analyse de ce jeu par la théorie des jeux donne comme solution l'équilibre de NASH, qui est ici la combinaison stratégique (AA_2,JW_1). Cette issue du jeu constitutif est un équilibre de NASH parce que AA_2 est la meilleure réponse de AA à JW_1 et que réciproquement, JW_1 est la meilleure réponse de JW à AA_2. En étudiant les quatre issues du jeu, on s'aperçoit que cette combinaison stratégique est la seule à être en équilibre de NASH. Si cette combinaison est la solution préconisée par la théorie, pour le jeu répété elle aurait dû être une des solutions.

Or, lors de l'expérience la combinaison stratégique la plus fréquente a été (AA_1,JW_2), utilisée dans 60% des cas, contre 14% des cas pour l'issue en équilibre de NASH.

On peut remarquer que l'issue la plus fréquemment utilisée est singulière puisqu'elle est PARETO-optimale. Dans le jeu constitutif, la combinaison (AA_1,JW_2) est PARETO-optimale car il n'existe aucune autre issue préférée par les deux joueurs.

En elle même l'expérience ne prouve pas grand chose, du fait de sa faible représentativité. Elle a cependant introduit un jeu nouveau, simple qui permet de modéliser une différence flagrante entre l'intérêt individuel (l'équilibre de NASH) et l'intérêt collectif (la PARETO optimalité). FLOOD et DRESHER ont demandé à NASH de faire des remarques sur cette expérience. Il leur a répondu en remarquant que les résultats sont fortement liés aux conditions de l'expérience. D'après lui si les choix avaient été faits à chaque étape sans pouvoir prendre en compte les choix précédents alors les résultats auraient été beaucoup plus proches de ceux obtenus grâce à la théorie. Ses idées sur la question confirment en quelque sorte les critiques initiales sur la théorie des jeux, à savoir son application difficile.

2.1.3 Les prisonniers

Cette expérience a jeté un trouble certain dans les esprits des différents chercheurs alors présents à la RAND Corp, et bien au delà. Certains comme VON NEUMANN ne l'ont pas prise au sérieux, et l'ont considérée comme une sorte de provocation. D'autres ont vu en ce jeu bien plus d'implications que de simples remises en cause des objectifs de la théorie des jeux.

C'est ainsi que DRESHER présenta le jeu à Albert TUCKER, mathématicien américain de l'université de Princeton, qui fut l'un des professeurs de NASH. Connaissant parfaitement à la fois VON

Neumann, Nash et la théorie des jeux, il a été invité par le département de psychologie de l'université de Stanford à donner un cours de théorie des jeux. La petite expérience de Flood et Dresher lui paraissant très intéressante, il se permit de la modifier, et de la présenter sous la forme d'une histoire de façon à pouvoir plus aisément être comprise par les étudiants. C'est cette histoire qui donna le nom au jeu : dilemme du prisonnier.

Tucker présente alors le jeu à ses étudiants de la manière suivante, que l'on peut lire dans [Pou92] :

> « *Two men, charged with a joint violation of law, are held separately by the police. Each is told that:*
> *(1) if one confesses and the other does not, the former will be given a reward, and the latter will be fined.*
> *(2) if both confesses, each will be fined.*
> *At the same time, each has good reason to believe that*
> *(3) if neither confesses, both will be clear.* »

Cette histoire s'est alors très vite répandue dans la communauté scientifique aussi bien chez les économistes, les mathématiciens, que les psychologues ou les spécialistes de science politique.

Le jeu simple a vite, et souvent, été caractérisé par une représentation stratégique telle que celle du tableau 2.2(a), dans laquelle le paiement de chaque joueur correspond en fait au nombre d'années de prison que le suspect va devoir faire.

	Coopérer	**Trahir**
Coopérer	(1 an, 1 an)	(3 ans, 0 an)
Trahir	(0 an, 3 ans)	(2 ans, 2 ans)

(a)

	`C`	`D`
`C`	(R, R)	(S, T)
`D`	(T, S)	(P, P)

(b)

Tab. 2.2 – *Le dilemme du prisonnier de* Tucker

Les deux stratégies disponibles sont alors :

- la *coopération*, que l'on notera `C` dans la suite, et qui correspond, dans l'anecdote de Tucker, à l'action de ne pas dénoncer son complice.
- la *trahison*, que l'on notera `D` dans la suite[11], et qui correspond à l'action de dénoncer son compère.

D'une manière plus formelle, le dilemme du prisonnier est représenté par la matrice de gains du tableau 2.2(b). La valeur des gains attribués à chaque joueur n'est pas aussi importante que l'ordre que l'on peut établir entre ceux-ci. La convention la plus fréquemment utilisée est alors de nommer ces gains comme dans le tableau 2.2(b), à savoir :

- **R** pour la *Récompense* de la coopération mutuelle (Reward en anglais) ;
- **P** pour la *Punition* de la trahison mutuelle (Punishment en anglais) ;
- **S** pour le *Salaire* de la duperie (Sucker's payoff en anglais) ;
- **T** pour la *Tentation* de la trahison (Temptation en anglais).

En considérant que le but du jeu est de maximiser son utilité pour que le jeu soit un dilemme du prisonnier il suffit que :

$$T > R > P > S \tag{2.1}$$

Dans le cas de la matrice du tableau 2.2(a) l'objectif de chaque joueur est d'obtenir le moins d'années de prison possible. Pour respecter les hypothèses de maximisation d'utilité il nous faut donc adapter la matrice en changeant les années de prison, en *années de liberté* ou bien changer l'ordre

11. Trahison se dit « *Defection* » en anglais.

de l'inéquation 2.1. On peut vérifier qu'avec l'une comme l'autre de ces adaptations le jeu représenté par la matrice du tableau. 2.2(a) est bien un dilemme du prisonnier : $T = 0$, $R = -1$, $P = -2$ et $S = -3$ pour l'adaptation en *années de liberté gagnées*, ou $T = 0$, $R = 1$, $P = 2$ et $S = 3$ pour le changement de sens de l'ordre donné par l'inéquation 2.1.

Il est à noter que cette convention ne correspond qu'au jeu constitutif du jeu répété utilisé pour l'expérience 5 de [Flo52], et que cette version est désormais symétrique en ce qui concerne les paiements des joueurs, ce qui en facilite grandement l'étude formelle.

Grâce à l'histoire des prisonniers, on peut aisément isoler le jeu simple, du jeu répété. Dans la suite de ce chapitre nous allons donc successivement étudier les caractéristiques de ces deux versions.

2.2 Le dilemme du prisonnier

2.2.1 Un jeu 2×2

Dans [RG66], Anatol Rapoport et Melvin Guyer ont classifié la totalité des jeux à deux joueurs et deux stratégies, qu'ils nomment jeux 2×2. Le nombre des joueurs étant fixé, il existe toujours une bijection évidente entre n'importe quel ensemble de deux joueurs et un ensemble prédéfini de joueurs, soit $N = \{N_1,N_2\}$. Le nombre de stratégies étant lui également fixé, le même argument peut être appliqué : l'ensemble des stratégies de base est alors $\{A_1,A_2,B_1,B_2\}$, et l'ensemble des combinaisons stratégiques est $S = \{(A_1,A_2),(A_1,B_2),(B_1,A_2),(B_1,B_2)\}$. Pour définir un jeu 2×2 il suffit donc de fournir les préférences de chaque joueur pour chaque issue, c'est-à-dire la fonction d'utilité. Les jeux 2×2 peuvent donc être tous définis sous forme stratégique par la donnée unique d'une fonction d'utilité.

Comme nous l'avons vu dans le chapitre précédent (page 11), seul l'ordre de préférences des issues d'un jeu est utilisé dans l'analyse théorique d'un jeu, et non l'amplitude de ces préférences. Pour simplifier leur étude Rapoport et Guyer ont énuméré toutes les fonctions d'utilités possibles pour de tels jeux en se fixant certaines hypothèses. Celles-ci, bien que fortes, ne violent pas l'idée de préférence à la base de la théorie des jeux, et permettent de comparer aisément les différents jeux.

Pour définir la fonction d'utilité ils se basent sur la matrice du tableau 2.3. Pour définir toutes les préférences possibles, il suffit de déterminer toutes les manières de classer les utilités a_i, b_i, c_i et d_i pour le joueur N_i, avec $i \in \{1,2\}$. Pour cela ils limitent les valeurs possibles pour les utilités aux entiers 1, 2, 3 et 4.

	A_2	B_2
A_1	a_1, a_2	b_1, b_2
B_1	c_1, c_2	d_1, d_2

Tab. 2.3 – *Matrice de base des jeux* 2×2.

A priori, le nombre de jeux pouvant être défini de cette manière est limité. Il y a $4! = 24$ ordres de préférences possibles pour N_1 et autant pour N_2, soit au total $24^2 = 576$ fonctions d'utilités possibles, et donc autant de jeux 2×2. Parmi tous ces jeux il existe cependant de nombreuses équivalences. Il suffit de penser à deux jeux identiques à un changement du rôle des joueurs prêts, ou à la manipulation des lignes et colonnes, c'est-à-dire du rôle des stratégies. En prenant en compte toutes ces équivalences, on aboutit à 78 jeux 2×2 différents.

Le travail de Rapoport et Guyer a permis de classifier ces 78 jeux. Leur taxonomie se base sur certaines propriétés qu'ils établissent en se basant sur les idées de la théorie des jeux. Parmi ces propriétés, on peut noter la notion d'équilibre à laquelle ils ajoutent des qualifications telles la vulnérabilité, la stabilité et la nature de cette stabilité. Le niveau le plus haut de leur taxonomie est appelé la classe du jeu :

1. Les deux joueurs ont une stratégie dominante
2. Un joueur seulement a une stratégie dominante

3. Aucun des deux joueurs n'a de stratégie dominante

Le niveau suivant permet d'utiliser les propriétés définies pour créer des sous-classes. Au final cette taxonomie compte 15 sous-classes. Une seule de ces sous-classes ne comportent qu'un seul jeu, c'est-à-dire un seul choix de préférences : la classe des jeux où les deux joueurs ont une stratégie dominante et qui ont un équilibre fortement stable mais non optimal. Le seul jeu qu'elle contient et le jeu numéro 12, représenté sur le tableau 2.4.

2, 2	4, 1
1, 4	3, 3

TAB. 2.4 – *Le jeu numéro 12 selon [RG66].*

Comme seul l'ordre des préférences est important on peut aisément déterminer les jeux équivalents au jeu 12 de [RG66] en déterminant cet ordre. Ceci est rendu d'autant plus facile que ce jeu est symétrique : les rôles des stratégies A_1 et A_2, ainsi que B_1 et B_2 sont identiques, c'est-à-dire que les utilités qu'ils rapportent aux joueurs sont identiques ($a_1 = a_2$, $d_1 = d_2$, $b_1 = c_1$ et $b_2 = c_2$). La préférence induite par le jeu pour $i \in \{1,2\}$ est

$$c_i < a_i < d_i < b_i$$

On reconnaît dans cette inéquation, à une inversion des stratégies près, l'ordre des préférences des issues qui caractérisent le dilemme du prisonnier selon TUCKER et qui est donné par l'inéquation 2.1.

2.2.2 Un modèle pour agents rationnels

RAPOPORT et GUYER caractérisent le dilemme comme le seul jeu 2×2 ayant un équilibre fortement stable mais non optimal. Cette non optimalité est une des critiques de la théorie des jeux.

La théorie des jeux, et plus généralement la théorie micro-économique moderne, est essentiellement basée sur le concept de rationalité individuelle. Les joueurs n'ont pour seul et unique objectif que la maximisation de leur utilité propre. L'augmentation de l'utilité collective n'est alors qu'un effet de bord du comportement individualiste des agents. Plus encore on considère que tous les agents sont rationnels. De la sorte, les modèles offerts par la théorie sont donc dédiés à n'étudier que des agents rationnels, rationalité et égoïsme ne faisant qu'un. Les groupes d'individus, ou collectivités ne sont donc en fait que des équilibres de comportements égoïstes. Laurent CORDONNIER, dans [Cor97], appelle cela la « Sainte Trinité » de la théorie micro-économique contemporaine : *rationalité-équilibre-économie.*

Le dilemme du prisonnier est souvent cité comme la faiblesse essentielle de cette approche de la rationalité. En effet si le comportement d'un groupe est une conséquence de comportements individuels égoïstes, alors la théorie doit montrer que les équilibres issus de ces comportements individuels favorisent aussi la collectivité. Les choix rationnels doivent bénéficier non seulement aux individus mais également à la collectivité.

Le dilemme du prisonnier montre justement le contraire. Sa singularité parmi l'ensemble des jeux les plus simples ne fait que renforcer les critiques de la théorie des jeux, et plus largement de la définition de la rationalité.

Le dilemme du prisonnier, dans sa version de base, est un jeu :

- **à deux joueurs**
- **à deux stratégies**
- **non coopératif**
- **non concurrentiel pur**
- **simultané**

– **symétrique**
– **à connaissance commune de rationalité**

Nous nommerons ce jeu le ***dilemme du prisonnier simple***.

Le fait qu'il n'y ait que deux joueurs et deux stratégies désigne le dilemme du prisonnier comme un des plus simples modèles de situation de conflits d'intérêts que l'on puisse imaginer. De nombreuses situations qui impliquent aussi bien des agents de même type, par exemple des humains, que de types différents, par exemple deux animaux de races différentes peuvent ainsi en être dérivées. Les deux stratégies représentent alors les deux comportements les plus basiques à savoir l'égoïsme ou l'altruisme.

La non coopérativité du jeu modélise le fait que les joueurs ne peuvent pas communiquer. L'utilisation de la communication, pour tenter de déterminer une solution par exemple, est souvent une des solutions les plus rapides en cas de conflit. Elle englobe un très large spectre de techniques de négociation qui va de la bataille à mort entre deux animaux jusqu'à la signature d'accords diplomatiques entre deux nations. Si l'on veut pouvoir étudier la résolution de conflits dans le cas le plus général il faut faire abstraction de cette communication. La simultanéité du jeu renforce encore plus nettement ce manque de communication. La relaxation de l'hypothèse de non coopérativité anihile le dilemme, une solution avec accord de partage des gains pouvant alors être atteinte rapidement

Le jeu n'est pas strictement compétitif de manière à pouvoir modéliser les situations de conflits d'intérêts dans lesquelles l'objectif n'est pas relatif mais absolu. L'intérêt n'est pas de battre son adversaire, auquel cas, de toute façon, aucune coopération ne serait alors possible, mais de renforcer sa propre condition.

La symétrie du jeu implique que les choix offerts aux deux agents ont exactement les mêmes conséquences pour l'un que pour l'autre, et assurent de ce fait que les deux intervenants n'aient que les choix les plus simples à leur disposition pour régler leur situation.

Enfin la connaissance commune de rationalité implique que les deux joueurs savent que leur adversaire est rationnel, comme eux.

L'étude formelle du dilemme du prisonnier dans ce cadre très précisément délimité de la théorie des jeux est donc, a priori, possible, et doit permettre de déterminer l'issue la plus *rationnelle*.

Le dilemme du prisonnier simple est donc le jeu DPS défini par :

– $J = \{A,B\}$ l'ensemble des joueurs
– $I = \{(\texttt{C},\texttt{C}),(\texttt{C},\texttt{D}),(\texttt{D},\texttt{C}),(\texttt{D},\texttt{D})\}$ l'ensemble des combinaisons stratégiques
– $V : I \to \mathbb{R}^2$ la fonction d'utilité définie par la matrice :

$$V : \begin{bmatrix} (R,\, R) & (S,\, T) \\ (T,\, S) & (P,\, P) \end{bmatrix} \tag{2.2}$$

et l'inéquation identique à l'inéquation 2.1 :

$$T > R > P > S \tag{2.3}$$

Il s'avère que le jeu possède un unique équilibre de Nash : la combinaison stratégique (D,D). En effet si l'on considère la matrice 2.2 et l'inéquation 2.3 comme la représentation stratégique du dilemme du prisonnier impliquant deux joueurs A et B, alors on s'aperçoit que pour chacun des deux joueurs la stratégie D domine strictement la stratégie C. Comme le demande la définition 1.13 (page 13) : $V(\texttt{D}|\texttt{D}) > V(\texttt{C}|\texttt{D})$, car $P > S$, et $V(\texttt{D}|\texttt{C}) > V(\texttt{C}|\texttt{C})$, car $T > R$. Les deux joueurs n'ayant que deux stratégies, D est donc la meilleure réponse de chacun des deux joueurs à l'autre. De ce fait (D,D) est donc la seule combinaison stratégique en équilibre de Nash. C'est donc la seule combinaison stratégique rationnelle, ce qui implique que la stratégie rationnelle est D, est que l'utilité espérée pour des agents rationnels est donc P : la punition. Il est clair que cette utilité est fortement sous-optimale, comme le note le classement de Rapoport et Guyer, puisque au moins deux autres issues offrent des utilités supérieures : R et T.

En poussant un peu plus l'étude du jeu on s'aperçoit que trois issues du jeu sont PARETO-optimales comme le définit la définition 1.12. Il y a quatre issues au jeu. Nommons ces quatre issues en fonction du vecteur d'utilité associé : a l'issue correspondant au vecteur (R,R), b celle correspondant au vecteur (S,T), c celle correspondant au vecteur (T,S) et enfin d celle correspondant au vecteur (P,P). Soit donc $I' = \{a,b,c,d\}$ l'ensemble des issues du jeu en bijection avec I. On obtient alors :

- $V_A(a) = R$, $V_B(a) = R$
- $V_A(b) = S$, $V_B(b) = T$
- $V_A(c) = T$, $V_B(c) = S$
- $V_A(d) = P$, $V_B(d) = P$

a est PARETO-optimale car :

- $V_A(b) < V_A(a)$ car $S < R$
- $V_B(c) < V_B(a)$ car $S < R$
- $V_A(d) < V_A(a)$ et $V_B(b) < V_B(a)$ car $P < R$

b est PARETO-optimale car B ne préfère aucune issue à b :

- $V_B(a) < V_B(b)$ car $R < T$
- $V_B(c) < V_B(b)$ car $S < T$
- $V_B(d) < V_B(b)$ car $P < T$

c est PARETO-optimale car A ne préfère aucune issue à c :

- $V_A(a) < V_A(c)$ car $R < T$
- $V_A(b) < V_A(c)$ car $S < T$
- $V_A(d) < V_A(c)$ car $P < T$

Dans chacune des ces trois issues il est impossible d'augmenter le gain d'un des joueurs sans baisser celui de l'autre. L'issue d, qui correspond au seul équilibre de NASH de DPS, n'est pas PARETO-optimale car il existe une issue préférée par les deux joueurs, à savoir l'issue a.

Il faut donc noter que dans le dilemme du prisonnier, l'équilibre de NASH ne correspond pas à une combinaison stratégique PARETO-optimale. C'est d'ailleurs la caractérisation du dilemme. C'est cette non coincidence entre l'intérêt individuel et l'intérêt collectif qui rend ce jeu si troublant et intéressant.

L'étude théorique de DPS qui ne prend en compte que les deux stratégies pures n'apporte rien de plus que ce qui vient d'être exposé. Une approche avec des stratégies mixtes en revanche permet de voir un peu plus précisément les choses.

2.2.3 Introduction de stratégies mixtes

Une stratégie mixte pour DPS est la donnée d'une distribution de probabilité $(p,1-p)$, avec $p \in [0,1]$ (p réel), sur l'ensemble des stratégies pures $\{\texttt{C},\texttt{D}\}$. Utiliser la stratégie mixte $(p,1-p)$ revient à jouer `C` avec une probabilité p et `D` avec une probabilité $1-p$. On représentera une telle stratégie uniquement par la probabilité p de jouer `C`. L'ensemble des stratégies mixtes pour le dilemme du prisonnier est donc un ensemble infini non-dénombrable. On peut remarquer que les deux stratégies pures `C` et `D` sont incluses dans l'ensemble des stratégies mixtes, puisqu'elles correspondent respectivement à $p = 1$ et $p = 0$.

L'ensemble des stratégies mixtes est infini, donc l'ensemble des issue est lui aussi infini, et il est impossible de décrire extensivement la valeur des gains associés à chaque issue. On peut cependant décrire la fonction d'utilité de manière précise, si l'on considère qu'elle correspond à l'espérance des gains. Si on considère que A joue la stratégie mixte p et que B joue la stratégie mixte q, alors l'espérance de gain de A est :

$$E(p|q) = (1-p)qT + pqR + (1-p)(1-q)P + p(1-q)S \tag{2.4}$$

Il devient alors aisé de représenter l'ensemble des espérances de gain par issues de DPS par une aire de $\mathbb{R}^2$ comme cela est fait sur la figure 2.1(a).

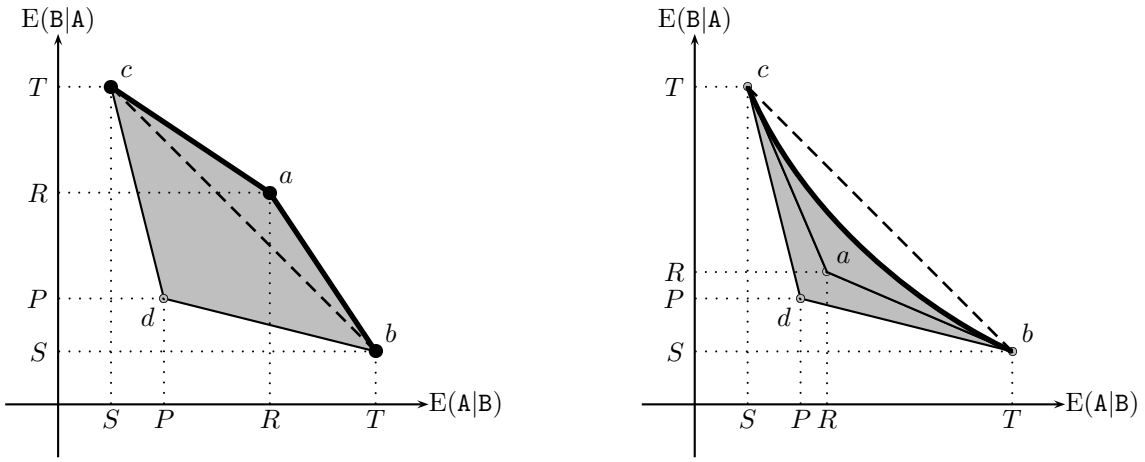


FIG. 2.1 – *Espérance de gain du dilemme du prisonnier avec stratégie mixte.*

On représente en abscisse les utilités espérées du joueur A, et en ordonnée les utilités espérées du joueur B. Toutes deux dépendent des stratégies p et q utilisées par A et B. Les aires grisées représentent donc toutes les issues possibles pour DPS avec des stratégies mixtes.

On peut facilement voir sur cette représentation où se trouve les issues qui sont PARETO-optimales. Il s'agit simplement de la surface délimitée par les issues c, a et b. En effet on comprend aisément sur cette représentation que n'importe quel point de cet surface limite est optimal car changer d'issue revient à plonger dans l'aire, donc à descendre les scores d'au moins un des deux joueurs.

Le jeu ne comporte toujours qu'un seul équilibre de NASH, qui est représenté sur la figure 2.1(a) par d. Il est clair que d n'est pas sur la surface de PARETO, et que donc l'équilibre exhibé par la théorie n'est pas optimal.

Sur la figure 2.1(a) les paramètres utilisés sont : $T = 6$, $R = 4$, $P = 2$ et $S = 1$. La figure 2.1(b) quant à elle représente un autre choix de paramètres, qui respecte toujours l'inéquation 2.3 : $T = 6$, $R = 2{,}5$, $P = 2$ et $S = 1$. On peut aisément remarquer que les deux aires de gains ne sont pas les mêmes, notamment en ce qui concerne la position de l'issue a. Dans le premier cas le point représentant cette issue se trouve au dessus du segment $[c,d]$, alors que dans le second cas il se trouve en dessous. Il est important de noter que dans le second cas l'issue a n'est plus PARETO-optimale. Il existe une infinité d'issues, c'est-à-dire de combinaisons stratégiques mixtes, offrant un gain supérieur à celui que peut fournir l'issue a.

Pour limiter l'étude du dilemme au cas le plus simple on se limitera aux cas de la classe de ceux de la figure 2.1(a), en nommant ***dilemme du prisonnier pur*** les jeux respectant l'inéquation 2.3 et l'inéquation :

$$2R > T + S \tag{2.5}$$

On peut trouver une justification plus précise de cette inéquation dans [Kuh99], et nous en retrouverons une autre avec la répétition du dilemme.

2.3 Le dilemme itéré du prisonnier

Dès son apparition le dilemme du prisonnier bien que simple [12] sème le trouble dans la communauté des théoriciens des jeux. Le problème majeur qu'il soulève est en fait la définition de la

12. SHUBIK écrit même *deceptively simple* dans [Shu70].

rationalité. Il semble cependant que très peu d'auteurs aient osé remettre en cause cette base de la théorie micro-économique définissant la rationalité comme l'intérêt individuel, ou l'égoïsme. Il est plus facile de trouver dans la littérature des *solutions* apportant des modifications à la théorie ou une approche différente du problème, dans le but de résoudre le dilemme.

L'apport le plus notable est celui fait par Martin Shubik dans [Shu70], qui reprend trois solutions différentes au dilemme du prisonnier.

2.3.1 Trois solutions au dilemme du prisonnier

Répétition infinie

La première solution proposée par Shubik, et qu'il attribue à Robert J. Aumann, est la plus simple. Dans cette solution il suffit de considérer que le jeu est répété une infinité de fois. On considère également que le gain d'un joueur est la moyenne de ses gains par jeu. On peut alors considérer que les joueurs ne prennent en compte que leur score moyen par jeu dans leurs décisions. De plus on peut désormais introduire la notion d'état stable comme étant toute combinaison stratégique rapportant en moyenne plus que l'issue d pour chacun des joueurs. N'importe lequel de ces états stables peut alors être appelé un *équilibre*, dans la mesure où si un des deux joueurs essaie de s'éloigner de cette position alors, l'autre peut le punir en utilisant le coup `D`, dans tous les coups suivants. On peut alors dire que les joueurs peuvent user de *menaces* crédibles, puisqu'elles coûteront à celui qui la subira plus que le gain espéré par la sortie d'un état stable. Si un tel équilibre est atteint aucun des deux joueurs n'a intérêt à jouer `D`, et donc à quitter cet équilibre, dans la mesure où dans ce cas il risque, en moyenne, de voir son score baisser.

Avec cette approche, il y a alors une infinité d'équilibres que l'on peut voir sur la figure 2.2. Tous les points de la zone hachurée peuvent être considérés comme des états stables, et donc des équilibres.

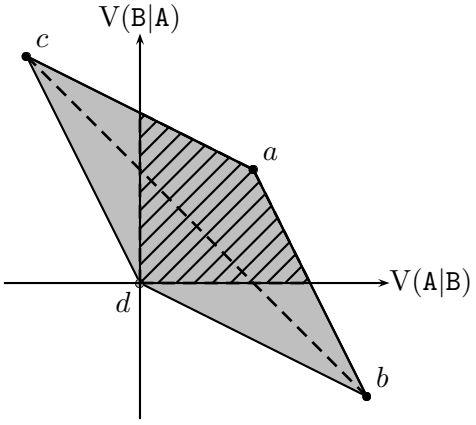


Fig. 2.2 – *Les gains en moyenne du dilemme du prisonnier répété à l'infini*

En effet toutes les issues de la zone hachurée correspondent à des états dans lesquels chacun des deux joueurs obtient en moyenne plus que P. Géométriquement il s'agit donc bien du quart nord-est d'un repère orthogonal dont d est l'origine.

Le point a fait partie de cet ensemble d'équilibres. Ce point correspond à une partie dans laquelle les deux joueurs jouent infiniment `C`. On comprend bien que si lors du premier coup les deux joueurs ont joué `C`, alors par la suite si chacun pense qu'un changement de comportement, à savoir jouer `D` une fois, peut entraîner une punition infinie par l'adversaire, alors les deux joueurs ne vont jamais quitter le status quo, et jouer `C` infiniment. La solution atteinte est alors optimale.

Le problème est que toutes les issues satisfaisant la rationalité individuelle sont en équilibre, comme on le voit sur la figure 2.2. Il y a donc un grand nombre de solutions sous-optimales à commencer par le point d qui reste un équilibre.

Cette solution de répétition infinie du dilemme n'est donc pas aussi forte que l'on pourrait l'espérer.

Jeux de survie sociale

Shubik définit une classe de jeu particulière qu'il nomme les *jeux de survie sociale* dans [Shu64]. Pour lui les jeux ne doivent pas prendre en compte uniquement le gain des joueurs. En fait il donnent une autre définition de la rationalité comme intérêt individuel. Pour lui la rationalité individuelle doit prendre en compte les gains des joueurs, *i.e.* les préférences données à chacune des issues d'un jeu, mais aussi la nécessité de survie. L'idée est que l'intérêt d'un joueur n'est pas seulement d'augmenter son capital, mais aussi de rester en vie afin de pouvoir profiter de son capital. La rationalité reste donc une idée profondément égoïste, même si elle induit des aspects plus globaux.

Pour prendre en compte cette idée il veut qu'au moins une de ces trois idées soient prises en compte dans les jeux :

1. les gains du présent sont plus important que les gains du futurs ;
2. il doit y avoir une probabilité, indépendante des gains, que le jeu se termine après une certaine durée, un certain nombre de répétitions ;
3. quand le jeu se termine les joueurs sont ruinés.

Dans le cas du dilemme du prisonnier il propose, dans [Shu70], deux aménagements très semblables. Le premier de ces deux aménagement consiste à considérer que le jeu est répété un nombre infini de fois mais que les gains sont escomptés par un facteur, qu'Axelrod appelera l'*ombre du futur*, et que l'on notera ω. Avec cet aménagement, qui correspond à l'introduction de l'idée 1, si l'on considère que $\mathrm{V}_t(\mathtt{A})$ est le score de la stratégie `A` pour le coup t alors le score total de `A` est :

$$\sum_{t=1}^{\infty} \omega^t V_t(\mathtt{A}) \tag{2.6}$$

Le second aménagement, correspondant à l'introduction de l'idée 2 considère qu'après chaque coup le jeu a une probabilité ρ de continuer, et donc $1-\rho$ de s'arrêter. Dans cette version les jeux peuvent donc être aussi bien finis qu'infinis, et les scores des joueurs se calculent alors de la même manière que pour l'aménagement précédent.

Sous ces nouvelles hypothèses Shubik montre qu'il est possible de faire un nouveau raisonnement sur les menaces. Désormais il est possible de déterminer des équilibres dans le jeu répété dépendant des valeurs utilisées pour ω. En fait il considère que les joueurs peuvent en quelque sorte prendre en compte un degré de vraissemblance pour une menace, et que ce degré est intimement lié au poids que le futur fait peser sur le présent. Il donne alors l'exemple d'une stratégie (voir page 72) qui permet d'obtenir un équilibre où la coopération est plus payante que la trahison, c'est-à-dire un exemple autre que la répétition de la stratégie d'équilibre (`D`) du dilemme simple.

Pour lui cette approche, qu'il nomme un peu rapidement *solution* du jeu, est plus un élargissement du modèle que l'ajout de propriété mathématique au jeu. En fait il considère vraiment que la rationalité individuelle est liée fortement à la survie de l'individu, et que donc être égoïste revient vraiment à essayer de prospérer pendant la durée de vie. Pour lui implicitement les joueurs n'ont donc pas intérêt à trahir longtemps, puisque plus le jeu avance moins la trahison rapporte en cas de punition par l'adversaire et de trahison mutuelle.

Même si la solution ne semble pas si triviale, c'est cette approche du jeu qui est la plus répandue dans la littérature depuis les travaux d'Axelrod. Nous y reviendrons par la suite.

Métajeux

La dernière solution présentée par Shubik dans [Shu70] est celle dite des méta-jeux. Elle est originellement présentée par N. Howard dans [How66].

Cette solution se base sur une version complètement différente du dilemme du prisonnier simple, qui ne considère pas uniquement deux stratégies, `C` et `D`, mais quatre, à savoir :

1. jouer `C`
2. jouer comme l'adversaire : jouer `C` (resp. `D`) si l'adversaire va jouer `C` (resp. `D`)
3. jouer le contraire de l'adversaire : jouer `C` (resp. `D`) si l'adversaire va jouer `D` (resp. `C`)
4. jouer `D`

Sous cette hypothèse la matrice du jeu n'est plus une matrice 2×2, mais une matrice 16×4. En effet le joueur `A` sachant que le joueur `B` a quatre choix possibles, en a lui même seize : quatre pour chaque choix possibles de son adversaire. En étudiant la matrice de ce jeu on se rend compte qu'il comporte trois équilibres de Nash. Deux de ces équilibres sont Pareto-optimaux, et le troisième est l'équilibre du dilemme simple.

À l'instar de Shubik et Axelrod nous pensons que ce genre d'artifices n'apportent pas grand chose à l'étude des situations modélisées par le dilemme du prisonnier car où les hypothèses utilisées affaiblissent beaucoup trop ouvertement celles utilisées dans le dilemme. On rappellera notamment que les joueurs ne peuvent pas communiquer. Il est donc impossible de pouvoir connaître, a priori, le coup que va jouer l'adversaire.

Les trois *solutions* exposées par Shubik ne sont en fait que des artifices permettant de montrer que la théorie des jeux peut apporter une solution *raisonnable* au dilemme du prisonnier, dont l'étude de base semble peu proche des comportements rencontrés sur des sujets humains.

La première information qui nous semble réellement intéressante dans ce résumé est que la notion de rationalité, au sens économique du terme, est mal définie ou en tout cas peu adéquate dans le cas de la modélisation de comportements humains.

La seconde est que la répétition du dilemme peut sous certaines conditions permettre d'obtenir des solutions optimales, c'est-à-dire permettant aux joueurs, même égoïstes, de coopérer. Binmore va même, dans [Bin99, section 7.5.4, page 310], jusqu'à dire :

> « *Ma propre opinion est que le dilemme du prisonnier n'est presque jamais un paradigme approprié pour traiter les problèmes de coopération qu'il est censé traduire. Le dilemme du prisonnier répété [...] est beaucoup plus pertinent dans ce rôle.* »

C'est à cette répétition du jeu que nous allons maintenant nous intéresser.

2.3.2 Répétition du dilemme

Il est maintenant clair que le dilemme du prisonnier de base est impropre à l'étude de situation de conflits dans lesquelles on espère voir apparaître des comportements coopératifs. La limitation principale du dilemme est incluse en partie dans le fait que le jeu est incapable de prendre en compte des comportements tels le non respect d'accord préalable, etc.

Le jeu est donc répété. Le jeu utilisé est défini sous les mêmes hypothèse que celles de la page 24. Une version répétée du dilemme du prisonnier sera appelé un *dilemme itéré du prisonnier*. Les hypothèses suivantes sont ajoutées :

- **les croyances des joueurs peuvent être modifiées au cours du jeu** ;
- **les joueurs ont la possibilité de voir les coups précédents de la partie**.

La première hypothèse est importante. Si on considère son contraire, à savoir que les croyances en matière de rationalité de l'adversaire ne peuvent en aucun cas être modifiées au cours du jeu répété, alors les résultats d'analyse du dilemme simple vont pouvoir s'appliquer directement. Dans ce cas la seule solution rationnelle est de toujours jouer `D`.

La seconde permet effectivement de réviser les croyances des joueurs au cours du temps. En effet si les joueurs ne peuvent pas voir les coups de l'histoire du jeu, la révision de leur croyance sur la rationalité de l'adversaire ne peut être qu'aléatoire et donc peu crédible.

Le jeu répété est basé sur le jeu constitutif DPS, tel que définit page 25. Un déroulement complet d'un jeu répété sera appelé une partie, ou encore un match (*game* ou *match* en anglais). Un jeu répété est constitué de plusieurs étapes, ou itérations. Chaque étape du jeu répété est constitué par un coup (*move* en anglais). Un coup correspond donc à une combinaison stratégique de DPS. Il y a donc quatre coups possibles à chaque étape d'un jeu [C,C], [C,D], [D,C] et [D,D]. À chaque étape d'une partie, c'est-à-dire pour chaque coup, un joueur utilise une des deux stratégies de DPS. Afin d'éviter la confusion entre stratégies pour DPS, et stratégies pour le dilemme répété, dans le cadre d'un jeu répété on dira qu'à chaque étape les joueurs choisissent une carte parmi les deux cartes C ou D. On réservera donc le mot de stratégie pour le comportement des joueurs pour le jeu complet, ici le jeu répété.

De façon à pouvoir étudier la coopération il faut qu'il soit plus intéressant pour les joueurs de toujours coopérer plutôt que successivement trahir puis coopérer. Cette restriction est modélisée dans le dilemme du prisonnier pur grâce au respect de l'inéquation 2.5. Cette inéquation obligeant $2R > T+S$, assure alors que les joueurs ont plus intérêt à coopérer tous les deux sur le long terme qu'à alternativement coopérer puis trahir, c'est-à-dire qu'à se faire exploiter et à exploiter. Nous étudierons dans le chapitre 6 une version du dilemme du prisonnier *non-pur* dans laquelle l'inéquation 2.5 n'est pas respectée.

Le dilemme itéré du prisonnier est un jeu répété et la recherche d'équilibres de Nash pour ce jeu a de grandes chances d'être fructueuse : il y en a un très grand nombre. La recherche d'équilibres parfaits en sous-jeux va, quant à elle, être plus difficile.

L'étude du dilemme répété dépend donc maintenant du nombre de répétitions. Plus exactement elle dépend de ce que les joueurs savent de ce nombre de répétition. Trois cas sont alors possibles :

1. Le nombre de répétitions est fini, et les joueurs ont connaissance de ce fait ;
2. Le nombre de répétitions est fini, mais les joueurs n'ont pas connaissance de ce fait ;
3. Le nombre de répétitions est infini et les joueurs ont connaissance de ce fait.

Dans le premier cas on parlera de dilemme répété à *horizon fini*, dans les autres cas on parlera de jeu à *horizon infini*. Nous allons maintenant étudier ces trois cas en détail.

2.3.3 Horizons

Répétition à horizon fini

Considérons donc un dilemme répété n fois. Considérons également que les joueurs ont connaissance du terme du jeu, c'est-à-dire le nombre exact de fois qu'ils vont se rencontrer. L'horizon du jeu est dit fini.

Dans ce cas la théorie des jeux avance le fait qu'il n'existe qu'un seul équilibre de Nash, et donc a fortiori un seul équilibre parfait en sous-jeux, à savoir la combinaison stratégique incluant uniquement la stratégie qui consiste à toujours trahir, c'est-à-dire toujours jouer D. Par la suite nous nommerons cette stratégie `all_d`.

La démonstration de l'unicité de cette *solution* se base sur le théorème de récurrence. On pourra en trouver une démonstration précise, par exemple, dans [Bin99, page 350] [13].

L'idée de la démonstration est relativement simple. À l'étape n les deux joueurs se retrouve face à un dilemme du prisonnier simple, exactement comme DPS. On considère en effet que les deux joueurs ne peuvent subir aucune menace crédible puisque le jeu est arrivé à sa dernière étape. Les joueurs ont donc tous les deux intérêt à trahir, et à jouer la stratégie de l'équilibre de Nash de DPS. Il est donc évident qu'au coup n les deux joueurs joueront D. Reste à savoir ce qu'ils ont à faire dans les $n-1$ premières étapes.

Sachant qu'au coup n le coup D va être joué par l'adversaire, aucune menace n'est plus crédible au coup $n-1$. On se retrouve alors confronté à un dilemme itéré du prisonnier répété $n-1$ fois. On

13. Il s'agit du théorème 8.3.1, dans lequel le coup D est appelé *Faucon*.

peut donc penser que les joueurs vont tenir le même raisonnement que précédemment. De ce fait ils vont jouer `D` au coup $n-1$.

En réappliquant ce raisonnement un nombre suffisant de fois, on arrive vite à la conclusion que le seul équilibre de Nash du jeu répété n fois est la combinaison stratégique (`all_d`,`all_d`), et que par conséquent la seule stratégie rationnelle est `all_d`.

Ce résultat implique que la coopération n'est pas rationnelle dans un dilemme répété un nombre fini de fois connu des joueurs.

En pratique, comme on peut par exemple le voir dans les résultats de l'expérience de Flood et Dresher dans [Flo52], la coopération est plus souvent utilisée que la trahison. Plus précisément comme le souligne Steven Kuhn dans [Kuh99] :

> « *In practice, there is not a great difference between how people behave in long fixed-length IPDs (except in the final few rounds) and those of indeterminate length.*
>
> *This suggests that some of the rationality and common knowledge assumptions used in the backwards induction argument (and elsewhere in game theory) are unrealistic.* »

Shubik, quant à lui, s'interroge plus profondément sur la théorie des jeux dans [Shu70] :

> « *The logical and silly conclusion is that even if the players played 1,000 times with a possible joint gain of $5,000 each they will end up with (0,0) having "out-psyched" each other thousand times!*
>
> *People do not play this way. [...] Can we match theory, experimental results and our casual observation of human affairs? Is all that is missing merely better mathematical analysis of the same model, or is a different model called for?* »

Même si le résultat théorique est indiscutable, il est clair qu'une fois de plus la théorie produit un résultat peu intéressant du point de vue de la modélisation de comportements humains.

Le cas des dilemmes du prisonnier itérés un nombre fixe de fois, avec ce nombre connu des joueurs, n'est donc pas un modèle intéressant de notre point de vue, et nous n'en parlerons donc plus.

Répétition à horizon infini

Considérons maintenant que les répétitions sont infinies, et que les joueurs savent que le jeu sera répété indéfiniment. Il est bien sûr évident qu'aucun humain ne peut avoir l'idée de jouer infiniment longtemps à un jeu, ne serait-ce que parce qu'il n'est pas immortel. De plus le nombre des stratégies alors utilisables est infini. Il devient alors difficile d'étudier le jeu.

Cependant si l'on restreint l'espace des stratégies à celles qui peuvent être représentées par des automates d'états finis, comme par exemple les automates de Moore, alors l'étude redevient possible. Dans ce cas le gain d'un joueur n'est pas la somme cumulée des gains de chaque coup joué, mais la moyenne des scores obtenus lors d'un cycle de l'automate.

On peut rappeler que si deux stratégies sont représentables par un automate de Moore alors une partie de taille infinie est en fait décomposable en deux phases, l'une initiale qui comporte un certain nombre de coups menant à une deuxième phase, que l'on appellera cycle, et qui se répétera indéfiniment. Cette propriété est due à la nature de l'automate, comme il est rappelé page 17.

La nature du gain d'un joueur, dans cette version itérée à l'infini, ne prend donc pas beaucoup en compte les premiers coups joués. Cette approche est une approche où l'horizon pour les joueurs est infini. L'argument de récurrence utilisé dans le cas précédent n'est plus utilisable, puisqu'il n'y a plus de dernier coup.

Malheureusement l'étude théorique de ce type de dilemme à horizon infini est difficile. En effet même dans le cas de deux stratégies automates, et même s'il est certain qu'un cycle sera atteint, il reste difficile de déterminer dans le cas général le meilleur comportement, ne serait-ce que parce que le calcul du cycle de coups répétés est lui même une tâche loin d'être aisée.

Dans la majorité des cas on préférera donc utiliser des dilemmes du prisonnier itérés un nombre fini de fois, mais dans lesquels les joueurs ne connaissent pas le nombre d'itérations. On dit souvent que ce sont des dilemmes du prisonnier à répétition indéfinie. Cela revient en quelque sorte au jeu de survie social de SHUBIK : on utilise un taux ω qui correspond soit à une réduction du poids des scores obtenus, de façon à donner de moins en moins d'importance aux coups futurs, soit à une probabilité que le coup en cours soit le dernier, que la prochaine étape n'existe pas. Techniquement il n'y a pas de différence. Ces genres de dilemmes sont également des dilemmes du prisonnier à horizon infini, puisque les joueurs n'ont aucune idée de la *date* de fin du jeu.

Dans la suite nous utiliserons la même notation que celle utilisée par AXELROD dans [AH81], repris dans [Axe84] [14], et [AD88]. Le facteur d'escompte sera appelé l'*ombre du futur* et noté ω, avec $0 < \omega < 1$. On identifiera souvent le joueur à sa stratégie. Le score d'un joueur utilisant une stratégie `A` contre un joueur utilisant une stratégie `B` sera noté $\mathrm{V}(\mathtt{A}|\mathtt{B})$. Le calcul de ce gain prendra donc en compte l'ombre du futur, de telle sorte que si on note $\mathrm{V}_n(\mathtt{A}|\mathtt{B})$ le gain de la stratégie `A` face à la stratégie `B` lors du coup n, alors :

$$V(\mathtt{A}|\mathtt{B}) = \sum_{i=0}^{\infty} \omega^i V_i(\mathtt{A}|\mathtt{B}) \tag{2.7}$$

Les dilemmes itérés du prisonnier à horizon infini comportent de très nombreux équilibres de NASH, comme nous l'avons vu dans les jeux de survie sociale de SHUBIK. Il est notamment possible d'avoir des équilibres optimaux au sens de PARETO. Certains de ces équilibres impliquent une coopération complète.

Il est intéressant de noter que la répétition du dilemme est finalement un meilleur modèle de l'étude de la coopération que le dilemme simple, et que dès le départ FLOOD et DRESHER avait donc bien cerné la difficulté d'une telle étude, et les problèmes que l'hypothèse de rationalité, sinon de définition de la rationalité, posent.

La prise en compte de l'horizon permet donc d'obtenir un modèle dans lequel la coopération, qui est un comportement a priori irrationnelle dans le dilemme simple, peut être considéré comme un choix rationnel. Reste à déterminer comment ce comportement peut être obtenu et dans quelle mesure il peut émerger dans une population d'agents a priori hétérogènes. En fait il est maintenant plus intéressant de s'intéresser à l'étude des comportements plutôt qu'à celui du modèle. C'est ce que nous ferons dans le reste de la thèse.

14. Ouvrage traduit en français en [Axe92, Axe96]

Chapitre 3

Stratégies et évaluations

Désormais, nous avons un modèle d'étude du comportement coopératif d'une population d'agents : le dilemme itéré du prisonnier à horizon infini. Nous pouvons alors décrire et étudier quelques comportements à travers ce modèle.

Dans ce but, nous décrivons dans un premier temps, un certain nombre de comportements classiquement répandus dans la littérature et d'autres plus originaux. Ensuite nous présentons les deux méthodes d'évaluation les plus utilisées dans le contexte du dilemme itéré du prisonnier et des simulations informatiques, que nous introduisons grâce aux travaux de Robert Axelrod, notamment présentés dans [Axe84], mais aussi de Maynard Smith, [Smi82]. Nous décrivons en quoi nos travaux diffèrent de ceux-ci. Pour finir, nous exhibons quelques propositions nouvelles sur cette *théorie de la coopération*.

3.1 Stratégies et comportements

3.1.1 Terminologie

Le dilemme itéré du prisonnier simple est un jeu à une étape. Deux stratégies seulement sont disponibles. On nomme ces deux stratégies, la coopération et la trahison, mais toute autre appellation peut être utilisée. On dit souvent être gentil ou être méchant, partager ou ne pas partager, etc. Tout n'est donc qu'une question de vocabulaire.

Les connotations habituelles que l'on retrouve dans le langage courant associées à ce vocabulaire gardent donc souvent leur sens. Cet état de fait peut être reproché par certains esprits un peu formalistes. Afin d'éviter ces reproches, le choix de représentation des stratégies par des lettres, `C` et `D`, est souvent fait. Nous l'avons d'ailleurs fait dans le chapitre précédent.

Il n'est cependant pas complètement inintéressant d'utiliser de telles connotations si l'on remet le dilemme du prisonnier dans son contexte d'utilisation initial. Ce contexte est d'ailleurs celui de toute la théorie des jeux, et plus largement des sciences économiques :

« *étudier les comportements d'agents (humains) en interactions.* »

Certes la théorie des jeux est une théorie mathématique et il faut éviter le plus possible les approximations sémantiques hasardeuses. Mais si l'on considère que les stratégies dans un jeu sont la représentation formelle des comportements, alors il semble naturel d'interpréter les notations mathématiques en terme d'attitudes en utilisant un vocabulaire plus *humain*. La théorie doit pouvoir s'appliquer, à un moment ou à un autre et ses applications doivent la nourrir.

Dans le cas du dilemme itéré du prisonnier la situation est encore plus *critique*. En effet dans le cas du dilemme simple, les stratégies sont au nombre de deux alors que dans le cas du dilemme itéré à horizon infini le nombre de stratégies est a priori indénombrable. Il faut donc choisir un mode de représentation. Le choix de description via le vocabulaire courant et la description des comportements humains n'est donc pas moins acceptable que le choix, par exemple, de la description extensive de tous les cas qu'une stratégie peut rencontrer et de la carte à jouer pour chacun de ceux-ci.

Les représentations intensionnelles des stratégies sont préférables aux représentations par extension.

3.1.2 Représentations formelles

Rappelons que les stratégies utilisables au dilemme itéré du prisonnier sont en fait des fonctions de l'histoire d'une partie vers l'ensemble des stratégies du dilemme du prisonnier simple.

Le comportement d'un agent n'est que la répétition d'un certain nombre d'actes de base dans un certain ordre. L'ordonnancement de ces actes de base est dynamique au cours d'une partie, c'est-à-dire qu'il peut être modifié en fonction de la situation dans laquelle se trouve l'agent.

Dans le dilemme itéré du prisonnier, un agent est un joueur. Les actes de base sont au nombre de deux et appartiennent à l'ensemble {C,D}. La situation dans laquelle se trouve un agent, est définie par le gain qu'il a accumulé depuis le début de la partie. Le temps est découpé en unités de base, qui sont les coups du jeu, numérotés de 1 à l'infini : le monde est discret. À un moment donné, le gain d'un joueur est fonction des gains obtenus lors des différents coups du passé. Ces coups sont fonction des cartes jouées par les joueurs.

Le comportement d'un agent, *i.e.* la stratégie qu'utilise le joueur, est donc fonction du comportement passé de son adversaire, qui, en l'espèce, représente l'environnement de l'agent.

Nous sommes donc bien dans un modèle simplifié à l'extrême d'interaction entre agents.

Définition 3.1 *Une* **histoire** *d'une partie de dilemme itéré du prisonnier est une séquence finie d'éléments de* $\{C,D\} \times \{C,D\}$.

La longueur d'une histoire est simplement le nombre d'éléments de la séquence. H est l'ensemble de toutes les histoires, H_n est le sous-ensemble composé des histoires de longueur n de H.

Par convention on notera (CCD) la séquence de cartes C, C, D, et par [CCC,DDD] l'histoire de la partie au coup 4 d'un match entre une stratégie coopérant tout le temps, et une stratégie trahissant tout le temps.

Définition 3.2 *Une stratégie s est une fonction de H vers* {C,D}*, spécifiant après chaque histoire la carte à jouer.*

Nous n'allons pas plus loin dans la description formelle des stratégies, qui ne nous sera pas utile dans le reste de la thèse. Le lecteur intéressé par la description précise des stratégies pures, mixtes et de comportements dans le cadre des jeux répétés peut se reporter au chapitre 4 de [AH92].

3.1.3 Quelques exemples

Les stratégies que nous décrivons ici, sont toutes des stratégies qui sont utilisables dans le simulateur logiciel servant de support aux expériences de la thèse. Nous les décrivons ici en utilisant le nom que le simulateur leur donne. C'est sous ce nom que nous les citerons par la suite.

Il ne nous faut bien entendu pas oublier que les stratégies que nous décrivons, utilisons, et étudions sont des représentations de comportements. C'est dans cet esprit que nous allons en décrire quelques-unes.

Afin de simplifier la compréhension de l'attitude à adopter par un joueur utilisant les stratégies décrites, nous utilisons la première personne du singulier lors de leur explication.

Nous présentons d'abord des stratégies pures.

Quelques comportements naïfs

Nous commençons l'énumération par quelques stratégies simples, correspondant à des comportements naïfs.

`all_c`

Je coopère toujours, quel que soit le comportement de mon adversaire.

Cette stratégie est souvent nommée `gentille`.

`all_d`

Je trahis toujours, quel que soit le comportement de mon adversaire.

Cette stratégie est souvent nommée `méchante`.

`per_ccd`

Je coopère puis je coopère puis je trahis, puis je coopère, puis je coopère puis je trahis, puis ...

Cette dernière stratégie fait partie des stratégies qui seront dites périodiques. Elles jouent cycliquement la même suite de cartes. Cette suite est appelée la période. Ici la période est composée de deux coopérations suivie d'une trahison. On note ce comportement : $(\texttt{CCD})*$. Un nombre infini de stratégies périodiques peut être décrit. Nous nous contenterons de celle-ci [15].

Introduction de la réactivité

Les stratégies qui suivent sont réactives. Dans au moins un cas, elles se comportent différemment après un changement de comportement de l'adversaire.

`spiteful`

Je coopère jusqu'à ce que mon adversaire ait trahi, après quoi je trahis toujours quel que soit son comportement suivant.

Cette stratégie est souvent nommée `rancunière`, `trigger`, ou encore `grim`.

`easy_go`

Je trahis jusqu'à ce que mon adversaire ait coopéré, après quoi je coopère toujours quel que soit son comportement suivant.

`tit_for_tat`

Je coopère au premier coup, ensuite à chaque coup, je joue la carte jouée par mon adversaire au coup précédent.

Cette stratégie est souvent nommée `donnant_donnant`, `oeil_pour_oeil`, ou surnommée `tft`. Elle a été créée par Anatol Rapoport lors du concours organisée par Robert Axelrod, dont nous allons parler dans la section suivante.

`mistrust`

Je trahis au premier coup, ensuite à chaque coup je joue la carte jouée par mon adversaire au coup précédent.

Cette stratégie est parfois nommée `méfiante`, ou `supiscious_tit_for_tat`.

De nombreuses variantes de stratégies basées sur les principes utilisés par ces deux dernières stratégies sont citées dans la littérature. Les trois suivantes en sont des exemples.

`two_tit_for_tat`

Je coopère mais trahis deux fois de suite après chaque trahison de mon adversaire.

`tf2t`

Je coopère sauf si mon adversaire a trahi deux fois consécutivement.

Le nom complet de cette stratégie est `tit_for_two_tat`.

15. Le simulateur logiciel en compte d'autres par défaut, et n'importe quelle stratégie périodique peut lui y être ajoutée simplement.

`slow_tft`

Je coopère deux fois, puis je coopère toujours sauf si mon adversaire a trahi dans l'un des deux derniers coups.

`soft_majo`

Je joue la carte que mon adversaire a majoritairement jouée dans l'histoire passée de la partie. S'il a joué autant de fois `C` que `D`, alors je coopère.

Cette stratégie est souvent nommée `majo_mou`.

Les stratégies utilisant ce principe de prise de décision en fonction du coup joué le plus souvent par l'adversaire sont dites *majoritaires*.

`pavlov`

Je coopère au premier coup, puis je coopère uniquement si mon adversaire et moi avons joué la même carte au coup précédent, sinon je trahi.

Cette stratégie a été introduite et étudiée par Kraines et Kraines dans [KK89] puis Nowak et Sigmund dans [NS93]. Elle est aussi nommée `P1`, ou encore `simpleton` [16] par Rapoport et Chammah qui l'ont découverte dans [RC65].

`prober`

Je commence par jouer la séquence (`DCC`), puis si mon adversaire a coopéré aux coups 2 et 3 alors je joue comme `all_d`, sinon comme `tit_for_tat`.

Cette stratégie est souvent appelée `sondeur`.

L'idée de base de la stratégie précédente est de tester son adversaire dès le début de la partie et d'adapter son comportement à celui de l'adversaire. Les stratégies utilisant ce principe sont appelées des *sondeuses* [17].

Nous décrivons maintenant quelques stratégies de comportement, que nous nommons plus fréquemment des stratégies probabilistes, du fait qu'elles font appel au hasard.

Les probabilistes

`ipd_random`

Je coopère et trahis avec une probabilité de 0,5 à chaque coup.

Ce comportement peut se résumer au lancer d'une pièce de monnaie, en attachant la décision du jeu de la carte `C` à une des faces et de la carte `D` à l'autre.

Cette stratégie est parfois nommée `lunatique`, `random` ou encore `hasard`.

Une longue série de stratégies probabilistes peut ainsi être définie. Chaque stratégie est alors parfaitement décrite par la probabilité p de coopérer [18]. La probabilité de trahir est alors $1 - p$.

Cette idée de probabilité, et d'intervention du hasard dans le comportement d'un individu peut être adaptée aux différentes stratégies que nous avons décrites précédemment. Ainsi la stratégie suivante est une adaptation probabiliste de `tit_for_tat`.

`hard_joss`

Je coopère au premier coup, ensuite si l'adversaire a trahi au coup précédent alors je trahis sinon je trahis avec une probabilité de 10%, et coopère avec une probabilité de 90%.

Cette stratégie est appelée simplement `joss` dans [Axe84].

16. qui signifie *nigaud* en anglais
17. Le simulateur en propose quelques unes par défaut
18. Une fois de plus le simulateur en propose un certain nombre.

D'autres comportements probabilistes sont envisageables. En effet, l'appel au hasard peut se faire en conjonction avec d'autres paramètres, comme le numéro du coup par exemple. Considérons que ce dernier est nommé `TURN`.

`worse_and_worse`

Je trahis avec une probabilité égale à `TURN`/1000, c'est-à-dire de plus en plus.

Nous décrivons maintenant simplement le comportement de deux stratégies particulières sur lesquelles nous reviendrons longuement dans le chapitre 4.

Deux stratégies particulières

`gradual`

Je coopère jusqu'à la première trahison de mon adversaire. Ensuite pour chaque trahison, je punis mon adversaire par n trahison(s), le calme par 2 coopérations et continue à coopérer. n est le nombre de fois que mon adversaire m'a trahi.

Cette stratégie est souvent nommée `graduelle`.

Nous expliquerons ce comportement plus en détail dans le chapitre 4, et restons donc volontairement discret pour le moment. Nous avons introduit cette stratégie dans [BDM96].

`bad_bet`

Je coopère jusqu'à ce que mon adversaire me trahisse. Ensuite je joue successivement les stratégies `tit_for_tat`, `all_c`, `spiteful`, et `per_ccd`, pendant 4 coups chacune. Puis je joue pendant les 4 coups suivants la stratégie qui m'a rapporté le plus gros score pendant les 4 coups où je l'ai utilisée. Et je recommence ce choix tous les 4 coups.

Cette stratégie est également étudiée en détails dans le chapitre 4, nous ne nous attardons donc pas sur son cas plus longuement.

Nous ne décrivons pas d'autres stratégies, mais le lecteur interessé pourra consulter la description de la totalité des stratégies disponibles dans le simulateur logiciel dans l'annexe A.

Une ébauche de taxonomie

Toutes les stratégies que nous avons décrites peuvent être caractérisées par certains traits. Nous présentons maintenant certaines des caractéristiques importantes qui peuvent être utilisées dans la composition d'un comportement.

bienveillance Une stratégie est dite bienveillante quand elle ne prend jamais l'initiative de la trahison. On dit aussi souvent que c'est de la *gentillesse*.

agressivité Une stratégie est dite agressive quand, au contraire, elle prend l'initiative de la trahison. On dit aussi que c'est de la *méchanceté*.

réactivité Une stratégie est dite réactive quand elle change son comportement après un changement de comportement de son adversaire. Le changement de comportement de la stratégie peut intervenir à n'importe quel moment après celui de l'adversaire, et pas forcément au coup juste après.

indulgence Une stratégie réactive est dite indulgente quand sa réaction n'implique pas que son comportement soit définitivement modifié. Plus exactement la période de réaction ne doit pas être de taille infinie.

Le tableau 3.1 caractérise l'ensemble des stratégies que nous avons décrites.

Stratégies	Bienveillance	Agressivité	Réactivité	Indulgence	Probabiliste
all_c	X				
all_d		X			
per_ccd		X			
spiteful	X		X		
easy_go		X	X		
tit_for_tat	X		X	X	
mistrust		X	X	X	
two_tit_for_tat	X		X	X	
tf2t	X		X	X	
slow_tft	X		X	X	
pavlov	X		X	X	
prober		X	X		
ipd_random		X			X
hard_joss		X	X	X	X
worse_and_worse		X			X
gradual	X		X	X	
bad_bet	X		X	X	

TAB. 3.1 – *Caractérisation des stratégies exemples*

3.1.4 Les comportements dans la nature

Avant de présenter les méthodes d'évaluation des stratégies, nous exhibons quelques exemples réels d'utilisation des comportements que nous venons de décrire.

Dans [Poo95], Robert POOL, fait le point sur les techniques de théorie des jeux utilisées dans l'analyse des comportements d'animaux divers et variés. Il présente notamment les comportements de groupe de guppy face à des prédateurs. Cette étude menée par MILINSKI et DUGATKIN montre que ces petits poissons utilisent en fait un comportement semblable à `tit_for_tat`. En effet lorsqu'un groupe de guppy est abordé par un prédateur, certains individus s'approchent de lui afin de tester sa faim, et sa *dangerosité*. Tous les guppys ne le font pas. On considère que ceux qui restent en arrière trahissent parce que ceux qui sont devant permettent aux autres d'obtenir des informations sur le prédateur et pas eux. MILINSKI et DUGATKIN ont remarqué en faisant des expériences avec des jeux de miroirs, que les guppys avaient une tendance à ne pas s'approcher du prédateur si lors d'une approche précédente, d'autres guppys ne les avaient pas accompagnés. Les guppys ont un comportement semblables à `tit_for_tat`.

Un comportement très similaire est relevé par HEINHSOHN et PACKER dans [HP95], repris dans [Mor95] sur le comportement de lions défendant leur territoire face à l'intrusion de femelles étrangères. Il semble cependant que dans ce cas le comportement soit un peu plus étrange, et complexe à analyser. Il est intéressant de noter que HEINSOHN et PACKER ont conscience de la simplicité du dilemme et de la complexité des comportements qu'il peut engendrer. Ils écrivent d'ailleurs en conclusion de leur article :

> « *Individual behavior in contests between larger group may prove to be even more complex.* »

C'est un point de vue que nous partageons avec eux, et sur lequel nous reviendrons plus loin dans ce chapitre.

Beaucoup plus récemment, l'étude de TURNER et CHAO publiée dans [TC99] et reprise par NOWAK et SIGMUND dans [NS99b] et [NS99a], montre que de petites bactéries se retrouvent dans des cellules sous deux formes $\Phi H2$, qui fabriquent moins de protéines qu'une autre souche sauvage $\Phi 6$. En mesurant le facteur d'adaptation de chacunes des versions de ces bactéries, on s'aperçoit qu'elles sont exactement en situation de dilemme du prisonnier [19]. L'étude menée montre que l'évolution des bactéries laisse une place très importante à un comportement de type `all_d`.

19. Il se trouve qu'elles sont dans une situation du dilemme non pure, l'inéquation 2.5 n'est pas respectée.

	C	D
C	(R=3, R=3)	(S=0, T=5)
D	(T=5, S=0)	(P=1, P=1)

TAB. 3.2 – *Le dilemme itéré du prisonnier classique*

3.2 Évaluations de stratégies

Maintenant que nous possédons une bonne quantité de stratégies, l'étape naturelle suivante est de tenter de les comparer, de les évaluer.

3.2.1 Préliminaires

Avant de commencer à étudier les méthodes classiques d'évaluation, nous fixons les notions et le vocabulaire utilisés.

Définition 3.3 *Un* **panel** *de stratégies est un multi-ensemble de stratégies.*

Un panel de stratégies est donc simplement un ensemble de stratégies dans lequel une même stratégie peut apparaître plusieurs fois. Une illustration naturelle peut être de considérer qu'un panel est un ensemble d'agents. Chaque agent a un comportement particulier. Ce comportement est défini par une stratégie. Il se peut donc que deux agents aient le même comportement et donc la même stratégie. Si la seule propriété du système que l'on veut étudier est le comportement des agents, alors il est pratique de confondre l'ensemble d'agents au multi-ensemble de leurs stratégies.

On dit d'un jeu répété que c'est un dilemme itéré du prisonnier classique s'il peut être représenté par la matrice 2.2(b) (page 22), que les paramètres de celle-ci vérifient les inéquations 2.3 et 2.5, et qu'ils ont les valeurs données par le tableau 3.2.

3.2.2 Tournoi

La première évaluation possible d'une stratégie est la comparaison de son score à celui d'une autre stratégie. Plus une stratégie à un score élevé, mieux on la considère. Ceci appelle immédiatement un résultat classique important dans l'étude de la coopération au travers du dilemme itéré du prisonnier.

Théorème 3.1 *Aucune stratégie ne peut faire un score optimal face à n'importe quelle stratégie.*

Preuve.
Si une stratégie fait un score optimal face à n'importe quelle stratégie, alors elle doit entre autre faire un score optimal face à `all_d` *et face à* `spiteful`*.*
Obtenir un score optimal face à `all_d` *ne peut se concevoir qu'en jouant comme* `all_d`*, et donc notamment en jouant la carte* `D` *au premier coup. Si une telle stratégie existe alors elle doit jouer* `D` *au premier coup.*
Face à `spiteful` *le meilleur comportement à adopter est de toujours coopérer. En particulier, si une telle stratégie existe alors elle doit jouer* `C` *au premier coup.*
Aucune stratégie, pure, mixte ou de comportement, ne peut jouer au premier coup, de façon certaine, `C` *contre une stratégie et* `D` *contre une autre du fait de l'hypothèse de non coopérativité du jeu.* □

Cela nous amène à faire une autre remarque. Dans le dilemme itéré du prisonnier, l'objectif du joueur n'est pas de gagner la partie, mais de gagner le plus de points possibles. Le dilemme n'est pas un jeu strictement compétitif. On peut noter que, si cela était le cas, une stratégie optimale, c'est-à-dire qui ne perd jamais aucun match, existe : `all_d`. Elle ne perd aucun match car elle joue toujours la carte `D`, qui lui assure un score de P au pire est de T au mieux, alors que son adversaire est alors assuré d'un score de S au pire et de P au mieux. Les inéquations 2.3 assure donc `all_d` de

jamais perdre aucun de ses matchs. En revanche, elle ne maximise pas son gain en points. Cette non maximisation du gain est contraire à l'hypothèse de rationalité individuelle de la théorie des jeux.

Il semble évident que, pour évaluer des stratégies, il faut avant tout pouvoir évaluer leur score. Soit V(A|B) le score de la stratégie A face à la stratégie B au cours d'une partie. Pour évaluer les stratégies d'un panel donné, on peut imaginer faire la somme des scores de chaque stratégie de ce panel face à toutes les stratégies de ce panel.

On imagine avec cette méthode faire une sorte de championnat sportif entre stratégies, on appelle souvent cela un tournoi [20].

Plus formellement soit $\mathcal{S}$ un panel de n stratégies. Le score dans un tournoi d'une stratégie s_i de $\mathcal{S}$ est

$$V(s_i) = \sum_{j=1}^{j=n} V(s_i|s_j)$$

Il est important de noter que le score d'une stratégie dans un tel tournoi inclut le score de la stratégie jouant face à elle-même. Cette inclusion est importante pour l'objectivité de l'évaluation. On peut ainsi comparer aisément toutes les stratégies du panel, les classer et donc déterminer dans ce panel quelle est la meilleure stratégie. Sans cette prise en compte on pourrait tomber dans le travers selon lequel la stratégie alors considérée comme la meilleure pourrait avoir été déclarée meilleure justement grâce au fait qu'elle n'ait pas rencontré de comportement identique au sien.

Il semble donc évident que pour être efficace une stratégie doit bien se comporter face à elle-même. Cette première nécessité implique immédiatement que all_d n'est pas un comportement optimal. En effet face à elle-même all_d n'arrive à obtenir qu'un score en moyenne de P.

Nous avons considéré que l'évaluation des scores des stratégies était possible. Nous utilisons cependant un dilemme à horizon infini. Le score d'une stratégie face à une autre dans ce cas est donné par l'équation 2.7 (page 33).

Le facteur d'escompte, appelé l'*ombre du futur*, noté ω, correspond à la probabilité que la partie s'arrête lors d'un coup. Plutôt que de recourir à ce paramètre pour des calculs effectifs, nous préférons fixer une longueur de partie et nous assurer que les stratégies ne prennent pas en compte cette longueur dans leurs décisions. Sous ces hypothèses, nous ne considérons plus le score d'une stratégie face à une autre comme une somme pondérée par un coefficient réel, mais simplement comme une somme.

Pour s'assurer que cette opération ne perturbera pas l'étude des résultats du dilemme, nous avons recours au théorème suivant qui nous permet de fixer facilement des longueurs de partie en fonction de l'ombre du futur.

Théorème 3.2 *La longueur moyenne d'une partie, $\overline{L}$, entre deux joueurs est dépendante du paramètre ω, plus précisement : $\overline{L} = \frac{1}{1-\omega}$.*

Preuve.
Soit $\omega' = 1 - \omega$, la probabilité que la partie continue après un coup.

$$\begin{aligned}
\overline{L} &= 1\omega' + 2\omega'(1-\omega') + 3\omega'(1-\omega')^2 + \cdots \\
&= 1(1-\omega) + 2(1-\omega)\omega + 3(1-\omega)\omega^2 + \cdots \\
&= \sum_{i=1}^{\infty} i(1-\omega)\omega^{i-1} \\
&= (1-\omega)\sum_{i=1}^{\infty} i\omega^{i-1} \\
&= (1-\omega)\sum_{i=1}^{\infty} \left(\omega^i\right)'
\end{aligned}$$

20. *round-robin tournament* en anglais

$$
\begin{aligned}
&= (1-\omega)\left(\sum_{i=1}^{\infty}\omega^i\right)' \\
&= (1-\omega)\left(\frac{\omega}{1-\omega}\right)' \\
&= (1-\omega)\frac{(1-\omega)+\omega}{(1-\omega)^2} \\
&= (1-\omega)\frac{1}{(1-\omega)^2} \\
\overline{L} &= \frac{1}{1-\omega}
\end{aligned}
$$

□

Désormais nous considérerons donc que les parties sont de taille fixe L, et nous choisirons les valeurs de L en fonction des valeurs de ω. Le score d'une stratégie `A` face à une stratégie `B` sera alors simplement :

$$V(\texttt{A}|\texttt{B}) = \sum_{i=0}^{i=L} V_i(\texttt{A}|\texttt{B})$$

Dans ce cadre, on peut observer sur le tableau 3.3 le classement obtenu dans un tournoi impliquant trois des stratégies décrites au début du chapitre.

	`all_d`	`easy_go`	`ipd_random`	**Score**
`all_d`	1 000	4 996	3 017	9 013
`easy_go`	1	2 998	2 253	4 522
`ipd_random`	495	3 979	2 253	6 727

TAB. 3.3 – *Un exemple de tournoi avec des parties de 1000 coups*

Il est à noter que ce tournoi a été obtenu grâce au simulateur logiciel que nous présenterons en détail dans le chapitre 4. Les scores obtenus dans les parties impliquant la stratégie `ipd_random` sont des moyennes.

Calculer un tournoi sur un panel $\mathcal{S}$ de n stratégies revient donc d'abord à calculer la matrice d'un jeu à deux joueurs et n stratégies, puis à faire la somme des scores par ligne, pour obtenir le score de chaque stratégie.

L'inconvénient majeur de l'évaluation par des tournois est que les stratégies bonnes dans un tournoi ne sont que très rarement bonnes dans un autre tournoi : elles ne sont pas robustes à la modification de leur environnement. Pour exemple on peut voir le piètre comportement de `all_d` dans le tournoi représenté sur le tableau 3.4.

	`tit_for_tat`	`spiteful`	`all_d`	**Score**
`tit_for_tat`	3 000	3 000	999	6 999
`spiteful`	3 000	3 000	999	6 999
`all_d`	1 004	1 004	1 000	3 008

TAB. 3.4 – *Un autre exemple de tournoi avec des parties de 1000 coups*

Mêmes si les tournois sont un bon moyen de comparaison de stratégies, une évaluation sérieuse ne peut se contenter de ce manque de robustesse des résultats. C'est, entre autre, pour réparer ce problème que les évolutions écologiques sont utilisées préférentiellement.

3.2.3 Évolution écologique

Une seconde méthode d'évaluation des stratégies est de simuler les principes de la sélection naturelle : moins un individu est fort, moins il a de chance de survivre. Cette simplification des mécanismes d'évolution est utilisée ici dans le cadre d'évolution de populations d'agents utilisant les stratégies d'un panel.

L'idée est de faire en sorte que chaque agent choisisse une stratégie dans le panel et rencontre un à un tous les autres agents. Il y a donc plusieurs représentants d'une même stratégie. Une fois que toutes les rencontres ont eu lieu chacun est alors capable de connaître la *valeur* de la stratégie qu'il utilise en cumulant les scores obtenus face à tous les autres. Les agents ayant choisi une stratégie ramenant peu de points ont alors tendance à disparaître, alors que ceux ayant choisi une stratégie rapportant beaucoup de points vont au contraire *croître et multiplier*. Le cycle recommence alors et on peut voir fluctuer la représentation des différentes stratégies dans la population globale.

L'évolution est reproduite tant que ces représentations ne se sont pas stabilisées, c'est-à-dire tant qu'entre deux cycles elles évoluent.

Le mécanisme évolutionnaire utilisé ici est une sorte de reproduction parthénogénétique dans laquelle un individu seul peut se reproduire. La population est polymorphique dans le sens où chaque individu a un comportement différent. On applique ainsi à la théorie des jeux les principes biologiques de l'évolution, lui donnant du même coup une nouvelle utilité, non pas d'étude des comportements économiques humains, mais d'étude de la dynamique des populations d'animaux.

On pourra trouver dans [Smi82] des descriptions très précises des différents mécanismes évolutifs que l'on peut appliquer à la théorie des jeux. Cet ouvrage, référence dans ce qui est devenu depuis la théorie des jeux évolutionnaires, explique de manière claire et formelle, les applications que la théorie des jeux peut avoir en biologie, mais aussi ce que la théorie des jeux peut apporter à la biologie.

Maynard Smith présente par exemple le dilemme du prisonnier comme un cas particulier de la compétition entre la colombe et le faucon pour la possession d'un territoire.

Considérons deux animaux se disputant un territoire. L'animal qui obtient le territoire est satisfait de ses conditions de vie, et va donc produire 5 progénitures, alors que celui ne l'ayant pas eu va rester dans des conditions précaires et ne pourra alors en produire que 3. La valeur du territoire en tant que *fitness*, au sens de Darwin, est alors de $5 - 3 = 2$. Maintenant, si ces deux animaux sont d'espèces différentes, comme un faucon et une colombe, alors le sort est vite réglé et la colombe se reproduira moins vite que le faucon. Si ces deux animaux sont de même espèce alors deux cas vont se présenter. Deux colombes se partagent le territoire en bon terme, et de manière équitable, alors que deux faucons vont se battre continuellement pour l'appropriation de la moindre parcelle de territoire. Au bout du compte, les deux faucons vont aussi se partager le territoire, mais la perte de force due au combat va se répercuter sur leurs capacités de reproduction. Maynard Smith modélise finalement ce cas de figure par un jeu, identique dans les hypothèses au dilemme du prisonnier, dont la forme normale est donnée sur le tableau 3.5 (le jeu étant symétrique on ne donne que la matrice de gains d'un des deux animaux). Les deux stratégies `H` et `D`, correspondent au comportement du faucon (*Hawk* en anglais) et de la colombe (*Dove* en anglais). V correspond au gain de l'acquisition du territoire, et C correspond au coût de la bataille entre deux animaux.

	H	D
H	$\frac{(V-C)}{2}$	V
D	0	$\frac{V}{2}$

Tab. 3.5 – *Le jeu Hawk-Dove*

Il note E(A|I) le gain d'un joueur utilisant la stratégie A contre un joueur utilisant la stratégie I. Il suppose alors que les individus d'une population infinie sont soit des faucons soit des colombes, et que ces individus se reproduisent en proportion de leur fitness. Il étudie ensuite les évolutions de la population globale.

Cette approche est basée sur son expérience de biologiste. Son passage par la théorie des jeux lui permet cependant d'étudier plus précisément le comportement d'animaux et l'évolution de ce comportement.

Il est surprenant de voir que sa méthode d'étude est quasiment la même que celle que nous avons décrite, avec quelques nuances cependant. En effet, MAYNARD SMITH, comme la plupart des théoriciens de la biologie, fait la plupart de ses études dans des cas continus : la population est infinie, la représentation des populations se fait en pourcentage, il utilise des stratégies mixtes, etc. On trouve alors de très nombreux résultats sur la dynamique des populations dans ces cadres, grâce à l'utilisation d'équations différentielles ou du simplex. HOFBAUER et SIGMUND synthétisent d'ailleurs un grand nombre de ces résultats dans [HS88] et [HS98], qui sont deux éditions consécutives d'un même livre[21]. De nombreux résultats sur la stabilité des populations y sont expliqués en détail. On peut également s'essayer au formalisme parfait de [AH94, chapitre 28], qui décrit minutieusement et de manière concise les relations entre biologie évolutionnaire et théorie des jeux.

Malheureusement notre vue des systèmes multi-agents, et plus simplement des agents, nous fait renoncer à de tels cas théoriques. La population de nos agents ne sera jamais infinie, un agent est indivisible, et à un moment donné, il faut pouvoir lui dicter son comportement. C'est pourquoi contrairement aux biologistes, et aux théoriciens des jeux, nous restons volontairement dans un cadre discret, déterministe, et calculable. C'est le cadre idéal pour les simulations informatiques.

Nous utilisons donc un algorithme basé sur la même idée que celle de MAYNARD SMITH en y apportant les quelques adaptations nécessaires.

Tout d'abord, nous limitons les stratégies utilisables par les agents à celles d'un panel particulier. Soit $\mathcal{S}$ ce panel, et $|\mathcal{S}|$ la taille de ce panel. Les stratégies de $\mathcal{S}$ sont notées s_i avec $i < |\mathcal{S}|$.

On considère de plus que le nombre d'invidus est fixé, et invariable du début à la fin de l'évolution. Soit Π la taille de cette population.

On note $W_n(\texttt{A})$ le nombre d'agents utilisant la stratégie `A` à la génération n. Autrement dit $\mathrm{W}_i(\mathrm{A})$ est la taille de la sous-population d'agents utilisant `A`. On dit que c'est la taille de la population de `A`.

Pour toute valeur de n on a

$$\Pi = \sum_{i=1}^{i=|\mathcal{S}|} W_n(s_i)$$

Le score d'un individu utilisant la stratégie s_i à la génération n est $V_n(s_i)$ avec

$$V_n(s_i) \quad = \quad \left(\sum_{j=1}^{j=|\mathcal{S}|} W_n(s_j) V(s_i|s_j) \right) - V(s_i|s_i)$$

Un individu a pour score la somme des scores qu'il obtient en jouant contre tous les autres joueurs, sauf lui.

Finalement la population de toutes les stratégies s_i est redistribuée de manière proportionnelle au score par

$$W_{n+1}(s_i) \quad = \quad \alpha_n W_n(s_i) V_n(s_i)$$

α_n est le coefficient de normalisation de la population

$$\alpha_n \quad = \quad \frac{\Pi}{\sum_{j=1}^{j=|\mathcal{S}|} W_n(s_j) V_n(s_j)}$$

21. Ces deux éditions sont complémentaires, certaines parties de la première édition ayant été remplacées dans la seconde.

Pratiquement tous les calculs se font sur des nombres entiers, on se contente donc de travailler dans un espace discret. La seule exception étant pour la normalisation : on arrondit à l'entier le plus proche pour le calcul des nouvelles tailles de population.

Dans les évolutions écologiques de bases, on choisira pour tout i

$$W_0(s_i) = \frac{\Pi}{|\mathcal{S}|}$$

Les figures 3.1 et 3.2 représentent des évolutions écologiques impliquant les mêmes stratégies que celles utilisées dans les tournois du tableau 3.3 et 3.4. Les populations de départ sont de 100 individus par stratégies.

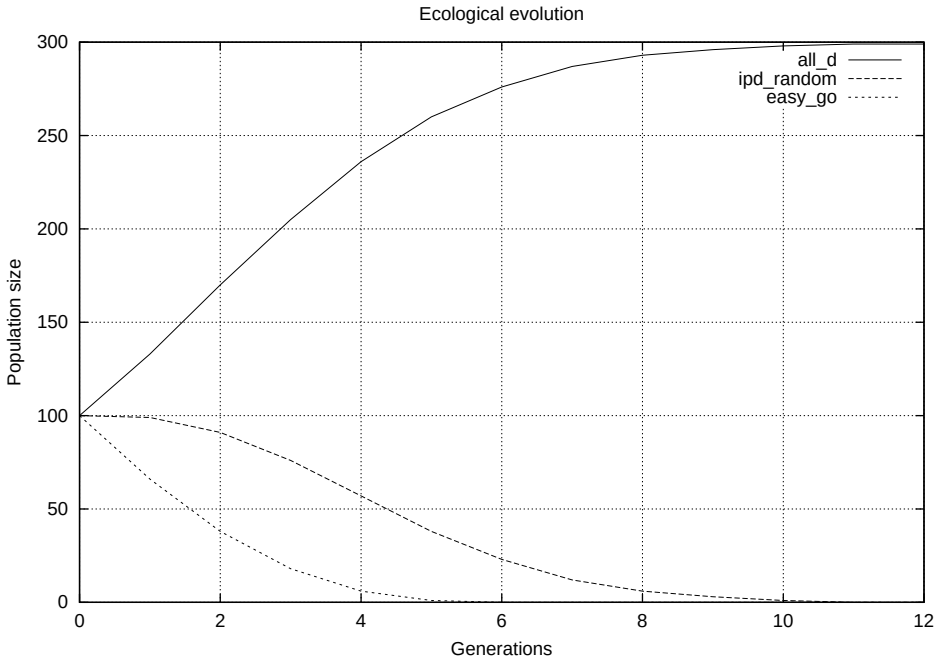


FIG. 3.1 – *Un exemple d'évolution écologique*

Ce genre d'évaluation pose évidemment beaucoup plus de questions que les tournois, comme par exemple la question essentielle de la stabilité des répartitions de populations.

3.2.4 Stabilité pour l'évolution

Dans [JG73], repris dans [Smi82], MAYNARD SMITH établit le concept de stratégies évolutionnairement stables. Ces stratégies sont des stratégies, si elles sont adoptées par tous les individus d'une population, qu'aucun mutant, c'est-à-dire un individu qui changerait subitement son comportement, ne peut envahir. Il applique cela au dilemme du prisonnier en établissant que `tit_for_tat` est une telle stratégie évolutionnaire sous certaines conditions sur la valeur d'ω.

AXELROD de son côté a défini le concept de *stabilité collective* d'une stratégie en modifiant légèrement le concept et la définition formelle de la stabilité évolutionnaire, et a montré que `tit_for_tat` est collectivement stable.

Puis BOYD et LOBERBAUM ont montré dans [BL87] qu'aucune stratégie pure ne pouvait être stable pour l'évolution.

En fait, les trois concepts très proches ne sont pourtant pas complètement identiques, et les trois auteurs ne discutaient en fait pas exactement de la même chose. Une série d'articles de BENDOR et SWISTAK, [BS96b, BS96a, BS95, BS97] a remis les choses en perspective. Nous reprenons ici l'excellent

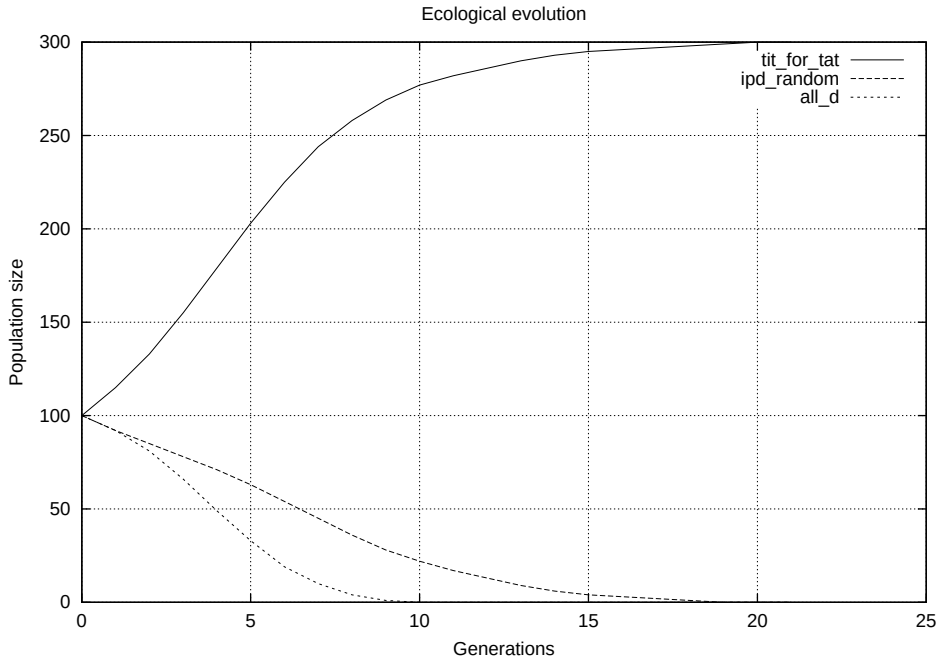


FIG. 3.2 – *Un autre exemple d'évolution écologique*

résumé de l'histoire de cette confusion, fait par KUHN dans [Kuh99], qui résume les différents concepts de stabilité dans les jeux évolutionnaires.

Être stable c'est en fait la capacité d'une population d'individus à résister à l'invasion par un étranger, ou un groupe d'étrangers. Les conditions de stabilité peuvent être qualifiées par trois propriétés.

Une condition est dite universelle si elle peut s'appliquer pour tous types de règles d'évolution, et réduite si elle ne s'applique qu'avec les dynamiques de reproduction [22].

On dit qu'une stabilité est forte quand la stratégie concernée permet d'éradiquer toute invasion, et faible quand elle permet seulement de survivre à une invasion.

Une condition est durable si elle s'applique aux invasions de n'importe quelle taille, et fragile si elle n'est utilisable qu'avec des invasions par des populations de petite taille.

Il existe en fait quatres hypothèses différentes pour qu'une stratégie soit stable pour l'évolution. Pour chacune d'elle on considère que V$(i|j)$ est le gain qu'un individu i obtient en rencontrant le joueur j.

La première est celle d'AXELROD, qu'il appelle la stabilité collective :

$$\forall j \ [V(i|i) \geq V(j|i)] \tag{3.1}$$

La seconde est celle de MAYNARD SMITH :

$$\forall j \ [V(i|i) > V(j|i) \text{ ou } (V(i|i) = V(j|i) \text{ et } V(i|j) > V(j|j))] \tag{3.2}$$

Les stabilités de ce type sont limitées, fortes et fragiles.

La troisième est celle de BOYD et LOBERBAUM :

$$\forall j \ [V(i|i) > V(j|i) \text{ ou } (V(i|i) = V(j|i) \text{ et } (\forall k \neq i)\,(V(i|k) > V(j|k)))] \tag{3.3}$$

Les stabilités de ce type sont universelles, faibles, et fragiles.

22. En anglais *replication dynamic*. On peut trouver des explications détaillées de cette dynamique dans [HS98], ou encore dans le chapitre 28 de [AH94]. Cette dynamique étant continue, nous n'en parlerons pas plus ici.

Enfin la dernière est celle de Bendor et Swistak :

$$\forall j \ [V(i|i) > V(j|i) \text{ ou } (V(i|i) = V(j|i) \text{ et } V(i|j) \geq V(j|j))] \tag{3.4}$$

Les stabilités de ce type sont limitées, faibles et fragiles.

Les stabilités au sens de Bendor et Swistak, et d'Axelrod peuvent être satisfaites par des stratégies du dilemme en évolution pouvant avoir un degré de coopération de 0 à 100%. En revanche les stabilités au sens de Maynard Smith ou Boyd et Loberbaum ne sont satisfaisables par aucune stratégie du dilemme en évolution à cause de l'existence de stratégies envahisseuses ayant le même comportement que la stratégie indigène face à elle-même, et que la stratégie indigène face à l'envahisseuse. Dans [BL87] l'exemple de l'invasion de `tit_for_tat` par `tf2t` est citée.

3.2.5 Quelques résultats

Dans une série d'articles repris dans [Axe84], Robert Axelrod rapporte les résultats de tournois et évolutions écologiques qu'il a menés grâce à des simulations informatiques. C'est, après ses travaux et la publication de son livre, que le dilemme du prisonnier a connu une très grande renommée. Les expérimentations d'Axelrod, mais aussi son immense talent de présentation du dilemme du prisonnier et surtout de ses applications et implications ont convaincu un grand nombre de l'intérêt de ce jeu comme modèle d'étude de la coopération. Depuis lors, l'intérêt soulevé par le dilemme du prisonnier n'a jamais faibli. On peut par exemple citer une bibliographie établie par Axelrod et D'Ambrosio pour la période de 1988 à 1994, dans laquelle ils citent plus de 200 références sur le dilemme du prisonnier. Depuis lors il n'y pas eu de signe flagrant de ralentissement : des numéros spéciaux de revues comme Biosystems par exemple, [Fog96], lui sont consacrés.

L'ouvrage d'Axelrod, bien que peu technique et formel, capture bien l'essence du dilemme du prisonnier et livre vraiment certaines clés sur l'étude de la coopération à tel point qu'il est souvent considéré comme le livre à la base d'une *théorie de la coopération*.

Les conclusions les plus importantes qu'Axelrod tire de ses travaux sont que pour être *bonne* une stratégie doit satisfaire quatre critères, [Axe84, page 110] :

> « *The advice takes the form of four simple suggestions for how to do well in a durable iterated Prisoner's Dilemma:*
>
> 1. *Don't be envious*
> 2. *Don't be the first to defect*
> 3. *Reciprocate both cooperation and defection*
> 4. *Don't be too clever* »

En bref, il conclut qu'une stratégie doit avoir au moins les quatre traits de caractères suivants pour espérer bien se comporter :

1. la gentillesse, ou encore la bienveillance
2. la réactivité
3. l'indulgence
4. la simplicité

Il exhibe la stratégie `tit_for_tat` comme exemple parfait de stratégie comportant ces caractéristiques. Il comfirme d'ailleurs ses opinions dans [Dav87], repris dans [Axe97], après avoir essayé les premiers algorithmes génétiques sur des stratégies à mémoire. Ces expériences semblaient le convaincre de la justesse de son analyse.

C'est le dernier point que nous remettons en cause. Pour nous, et les deux prochains chapitres en apporteront des arguments, la simplicité n'est pas un critère de qualité. Au contraire, nous pensons qu'il existe des complexifications infinies dans les comportements qui favorisent la coopération et son émergence.

D'ailleurs à la lecture des résultats de [Dav87], et en particulier des motifs exhibés par AXELROD sur les chromosomes des meilleures stratégies ayant évolué, rien ne laisse penser que la simplicité soit un des traits de caractère de ces stratégies. Moins sérieusement on peut dire qu'AXELROD nous donne raison en nommant son deuxième ouvrage traitant du dilemme du prisonnier, [Axe97] : « *The Complexity of Cooperation* ».

Il nous paraît surprenant que, dans un modèle où la communication est très difficile, la coopération ne soit favorisée que par des actes simples. La difficulté de communiquer, et donc de comprendre ou de se faire comprendre par son adversaire nous fait plutôt penser, qu'il faut savoir interpréter des comportements très différents, et que cette capacité n'est pas favorisée par la simplicité des comportements.

Certes, dans un environnement restreint, l'intérêt des agents peut être d'acquérir ce trait de caractère et donc de faire converger la population vers un comportement simple. Mais si l'environnement vient à changer alors la perte de complexité peut être fatale. S'habituer à la simplicité est forcément un handicap quand survient un événement extraordinaire.

Nous montrons, dans la section suivante, qu'il est cependant difficile, et même impossible d'affirmer définitivement notre point de vue sur la complexité, comme celui opposé sur la simplicité, en traitant le cas de l'améliorabilité infinie des panels de stratégies. La seule méthode pour défendre notre point de vue est alors l'accumulation d'arguments en sa faveur, ce que nous faisons dans les chapitres suivants.

3.3 Améliorabilité

La question à laquelle nous allons répondre ici est la suivante : « *est-il possible d'ajouter dans tout panel une stratégie meilleure que la meilleure du panel ?* »

3.3.1 Un exemple

Nous construisons ici un exemple de panel qui peut être amélioré à l'infini.

Définition 3.4 *La stratégie $\mathcal{A}_n$ trahit pendant les n premiers coups d'une partie puis coopère toujours ensuite.*

On note $\mathbb{A}_N$ l'ensemble des stratégies $\mathcal{A}_n$ avec $n < N$.

Théorème 3.3 *Dans un tournoi de dilemme itéré du prisonnier classique sur le panel $\mathbb{A}_N$ et pour toute valeur $N \in \mathbb{N}$, il est toujours possible de trouver une stratégie s telle que si cette stratégie est ajoutée au panel elle gagne le tournoi.*

Preuve.
Soit L la longueur des parties. Le score d'une stratégie $\mathcal{A}_i$ de $\mathbb{A}_N$ est $V(\mathcal{A}_i)$ avec :

$$\begin{aligned} V(\mathcal{A}_i) &= \sum_{j=1}^{j=i-1} (jP + (i-j)T + (L-i)R) \\ &\quad + iP + (L-i)R \\ &\quad + \sum_{j=i+1}^{j=N} (iP + (j-i)S + (L-j)R) \\ &= \sum_{j=1}^{j=i} (jP + (i-j)T + (L-i)R) \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{j=i+1}^{j=N} (iP + (j-i)S + (L-j)R) \\
= \;& \frac{1}{2}SN^2 + \frac{1}{2}Si^2 + \frac{1}{2}Ri - \frac{1}{2}iT - \frac{1}{2}RN \\
& - \frac{1}{2}RN^2 - \frac{1}{2}Si + \frac{1}{2}SN + RLN - \frac{1}{2}Ri^2 \\
& - \frac{1}{2}Pi^2 - SiN + PiN + \frac{1}{2}i^2T + \frac{1}{2}Pi
\end{aligned}
$$

Dans le cas du dilemme itéré du prisonnier classique cette équation se simplifie et devient :

$$V(\mathcal{A}_i) = -\frac{1}{2}i - \frac{3}{2}N - \frac{3}{2}N^2 + 3LN + \frac{1}{2}i^2 + Ni \tag{3.5}$$

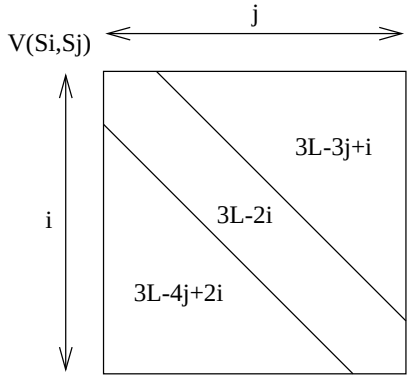
Comme $\frac{\delta V}{\delta i} = -\frac{1}{2} + i + N$ *est toujours positif, on peut dire que* V *est strictement croissante.*
Donc si on ajoute la stratégie $\mathcal{A}_{N+1}$ *à l'ensemble* $\mathbb{A}_N$*, alors* $\mathcal{A}_{N+1}$ *gagne le tournoi face à toutes les stratégies de* $\mathbb{A}_N$*.* □

Juste pour information, on peut noter que la différence entre deux ensembles de stratégies est :

$$V(\mathcal{A}_N) - V(\mathcal{A}_{N-1}) = 2N - 1$$

De plus on peut caractériser la matrice des gains du tournoi :

$$
\begin{aligned}
\forall\, 1 < i < N, \forall\, 1 < j < i,\ V(\mathcal{A}_i|\mathcal{A}_j) &= RL - j|R-P| + (i-j)|R-T| \\
&= 3L - 4j + 2i \\
\forall\, 1 < i < N,\ V(\mathcal{A}_i|\mathcal{A}_i) &= RL - i|R-P| \\
&= 3L - 2i \\
\forall\, 1 < i < N, \forall\, i < j < N,\ V(\mathcal{A}_i|\mathcal{A}_j) &= RL - i|R-P| + (j-i)|R-S| \\
&= 3L - 3j + i
\end{aligned}
$$



On peut vérifier ce résultat sur l'ensemble $\mathcal{A}_3$, avec des parties de longueur $L = 1000$:

$\mathcal{A}_1$:(D) + (C)∗ $\rightarrow V(\mathcal{A}_1|\mathcal{A}_2) = 1 + 0 + 3 + 3 + 2988 = 2995$
$\mathcal{A}_2$:(DD) + (C)∗ $\rightarrow V(\mathcal{A}_2|\mathcal{A}_1) = 1 + 5 + 3 + 3 + 2988 = 3000$

$\mathcal{A}_1$:(D) + (C)∗ $\rightarrow V(\mathcal{A}_1|\mathcal{A}_3) = 1 + 0 + 0 + 3 + 2988 = 2992$
$\mathcal{A}_3$:(DDD) + (C)∗ $\rightarrow V(\mathcal{A}_3|\mathcal{A}_1) = 1 + 5 + 5 + 3 + 2988 = 3002$

$\mathcal{A}_2$:(DD) + (C)∗ $\rightarrow V(\mathcal{A}_2|\mathcal{A}_3) = 1 + 1 + 0 + 3 + 2988 = 2993$
$\mathcal{A}_3$:(DDD) + (C)∗ $\rightarrow V(\mathcal{A}_3|\mathcal{A}_2) = 1 + 1 + 5 + 3 + 2988 = 2998$

3.3.2 Le cas général

Nous étudions maintenant les cas d'améliorabilité dans un cadre plus large.

Définitions

Définition 3.5 *On dira que s domine le panel M en tournoi limité (resp. en évolution écologique limitée), et on notera* $s >_t^n M$ *(resp.* $s >_e^n M$*), si s gagne le tournoi (resp. l'évolution écologique) sur le panel* $M \cup \{s\}$ *avec des rencontres limitées à n coups.*

Définition 3.6 *On dira que s domine le panel M en tournoi quelconque (resp. en évolution écologique quelconque), et on notera* $s >_t^\forall M$*, si* $\forall\ n, s >_t^n M$ *(resp.* $s >_e^\forall M$*, si* $\forall\ n, s >_e^n M$*)*

Définition 3.7 *On dira que s domine le panel M en tournoi (resp. en évolution écologique), et on notera* $s >_t M$ *si* $\exists n_0$ *tel que* $\forall n \geq n_0$*,* $s >_t^n M$ *(resp.* $s >_e M$ *si* $\exists n_0$ *tel que* $\forall n \geq n_0$*,* $s >_e^n M$*).*

Définition 3.8 *On dira alors qu'un panel M est :*

- $[t,n]$*-améliorable si* $\exists s,\ s >_t^n M$
- $[e,n]$*-améliorable si* $\exists s,\ s >_e^n M$
- $[t,\forall]$*-améliorable si* $\exists s,\ s >_t^\forall M$
- $[e,\forall]$*-améliorable si* $\exists s,\ s >_e^\forall M$
- $[t]$*-améliorable si* $\exists s,\ s >_t M$
- $[e]$*-améliorable si* $\exists s,\ s >_e M$

Les questions intéressantes sont donc celles qui concernent chacune des *améliorabilités* définies. La question générale étant :

Est-ce que tous les panels sont . . . -améliorables ?

Pour répondre à ces questions nous allons considérer que pour chacune d'elle la réponse est négative, et que donc il est possible de trouver ou construire un contre-exemple à chaque fois.

Les questions se ramènent donc à ces 6 questions :

1. Existe-t-il un panel qui ne soit pas $[t,n]$-améliorable ?
2. Existe-t-il un panel qui ne soit pas $[e,n]$-améliorable ?
3. Existe-t-il un panel qui ne soit pas $[t,\forall]$-améliorable ?
4. Existe-t-il un panel qui ne soit pas $[e,\forall]$-améliorable ?
5. Existe-t-il un panel qui ne soit pas $[t]$-améliorable ?
6. Existe-t-il un panel qui ne soit pas $[e]$-améliorable ?

Tous les exemples ne se font que sur des stratégies pures.

$[t,n]$-améliorabilité

Il est possible de construire pour chaque valeur de n un panel qui ne soit pas $[t,n]$-améliorable. Supposons n fixé, et étudions le panel M composé de m stratégies `all_d` [23] :

$$M = \{\underbrace{\texttt{all\_d}, \cdots, \texttt{all\_d}}_{m \text{ fois}}\}$$

23. Pour éviter d'avoir à utiliser m stratégies `all_d`, on peut créer artificiellement des stratégies qui sur les n premiers coups ont le même comportement. Par exemple : *trahir les* $n+i$ *premiers coups puis coopérer*. Pour simplifier nous considèrerons que les stratégies qui ont le même comportement sur les n premiers coups ont le même nom, même si elles ont un comportement différent sur les coups supérieurs à n.

En ajoutant s on ne peut l'avantager que lorsque qu'elle joue contre elle-même, donc on ne peut lui faire gagner, par rapport à `all_d` contre elle-même, que $(R-P)n$ points. Dans le cas classique cela revient à $2n$ points.
Pour cela il lui faut toujours coopérer contre elle-même, donc au mieux elle doit jouer une fois `C` contre `all_d`. Ceci lui fait perdre, toujours par rapport à `all_d` contre elle-même, $(P-S)m$ points. Dans le cas classique cela revient à perdre m points.
Si $(P-S)m > (R-P)n$ alors s ne peut pas être gagnante dans $M \cup \{s\}$.
Il suffit alors de choisir m tel que $m > \frac{(R-P)n}{(P-S)}$ pour que M ne soit pas $[t,n]$-améliorable.
Il existe un, et même une infinité de panels qui ne sont pas $[t,n]$-améliorable. □

$[e,n]$-améliorabilité

En utilisant la même idée que précédemment combinée au fait qu'une évolution écologique sur un tel panel n'implique que deux stratégies, dont une est la meilleure en tournoi, et obtient donc le meilleur score face à toutes les autres, il devient alors évident qu'il existe une infinité de panels qui ne sont pas $[e,n]$-améliorables. □

$[t,\forall]$-améliorabilité

En ce basant sur le résultat de la $[t,n]$-améliorabilité, on peut construire un panel non $[t,n]$-améliorable, qui est donc par hypothèse non $[t,\forall]$-améliorable. □

$[e,\forall]$-améliorabilité

En ce basant sur le résultat de la $[e,n]$-améliorabilité, on peut construire un panel non $[e,n]$-améliorable, qui est donc par hypothèse non $[e,\forall]$-améliorable. □

$[t]$-améliorabilité

Soit le panel M suivant :

$$M = \{\underbrace{\texttt{tft},\cdots,\texttt{tft}}_{m \text{ fois}}\}$$

Supposons que $s \neq \texttt{tft}$ et que $s >_t M$. Soit alors n_0 tel que $s >_t^n M$ pour tout $n \geq n_0$. Quand s joue contre `tft`, s trahit au moins une fois en premier, car sinon il serait impossible que $s >_t^n M$.
On note $V_i(A|B)$ le score de la stratégie A face à la stratégie B au coup i, et $V_i(A)$ le score de la stratégie A au coup i.
Soit p_0 le premier coup où s trahit contre `tft` :

	1	2	...	p_0-1	p_0
s	C	C	...	C	D
tft	C	C	...	C	C

	1	2	...	p_0-1	p_0
s	C	C	...	C	X
s	C	C	...	C	X

On peut noter que pour des stratégies pures, seul X=`D` est envisageable.
Jusqu'au coup p_0, `tft` et s font le même score :

$$\forall i < p_0,\ V_i(s|\texttt{tft}) = V_i(\texttt{tft}|s) = (m+1)R$$

Au coup p_0 s fait un plus gros score que `tft` :

$$\begin{aligned} V_{p_0}(s) &= V_{p_0}(s|\texttt{tft}) + V_{p_0}(s|s) \\ &= mT + P \\ &\leq mT + R \\ V_{p_0}(\texttt{tft}) &= V_{p_0}(\texttt{tft}|s) + V_{p_0}(\texttt{tft}|\texttt{tft}) \\ &= mR + S \end{aligned}$$

Étudions maintenant le coup $p_0 + 1$:

1. s **joue** `C` **contre** `tft` **au coup** $p_0 + 1$

	1	2	...	$p_0 - 1$	p_0	$p_0 + 1$
s	C	C	...	C	D	C
tft	C	C	...	C	C	D
	1	2	...	$p_0 - 1$	p_0	$p_0 + 1$
s	C	C	...	C	X	Y
s	C	C	...	C	X	Y

$$\begin{aligned} V_{p_0+1}(s) &= V_{p_0+1}(s|\texttt{tft}) + V_{p_0+1}(s|s) \\ &\leq mS + R \\ V_{p_0+1}(\texttt{tft}) &= V_{p_0+1}(\texttt{tft}|s) + V_{p_0+1}(\texttt{tft}|\texttt{tft}) \\ &= mR + T \end{aligned}$$

Finalement pour les coups p_0 et $p_0 + 1$, s gagne moins de $m(T + S) + 2R$ alors que `tft` gagne $2mR + S + T$. Or grâce aux inéquations 2.3 et 2.5 $2mR + S + T > m(T + S) + 2R$ se réduit en $m > 1$, donc si $m > 1$ on n'a alors pas $s >_t^{p_0+1}$ `tft`.
Donc si $m > 1$ on peut supposer que s va jouer `D` contre `tft` au coup $p_0 + 1$.

2. s **joue** `D` **contre** `tft` **au coup** $p_0 + 1$

	1	2	...	$p_0 - 1$	p_0	$p_0 + 1$
s	C	C	...	C	D	D
tft	C	C	...	C	C	D
	1	2	...	$p_0 - 1$	p_0	$p_0 + 1$
s	C	C	...	C	X	Y
s	C	C	...	C	X	Y

$$\begin{aligned} V_{p_0+1}(s) &= V_{p_0+1}(s|\texttt{tft}) + V_{p_0+1}(s|s) \\ &\leq mP + R \\ V_{p_0+1}(\texttt{tft}) &= V_{p_0+1}(\texttt{tft}|s) + V_{p_0+1}(\texttt{tft}|\texttt{tft}) \\ &= mR + P \end{aligned}$$

Finalement pour les coups p_0 et $p_0 + 1$, s gagne moins de $m(T + P) + 2R$ alors que `tft` gagne $2mR + S + P$.
Étudions alors le coup $p_0 + 2$:

(a) s **joue** `C` **contre** `tft` **au coup** $p_0 + 2$

	1	2	...	p_0-1	p_0	p_0+1	p_0+2
s	C	C	...	C	D	D	C
tft	C	C	...	C	C	D	D

	1	2	...	p_0-1	p_0	p_0+1	p_0+2
s	C	C	...	C	X	Y	Z
s	C	C	...	C	X	Y	Z

$$\begin{aligned} V_{p_0+2}(s) &= V_{p_0+2}(s|\texttt{tft}) + V_{p_0+2}(s|s) \\ &\leq mS + R \\ V_{p_0+2}(\texttt{tft}) &= V_{p_0+2}(\texttt{tft}|s) + V_{p_0+2}(\texttt{tft}|\texttt{tft}) \\ &= mR + T \end{aligned}$$

Finalement pour les coups p_0, p_0+1 et p_0+2, s gagne moins de $m(T+P+S)+3R$ alors que tft gagne $3mR+S+P+T$. Or toujours grâce aux inéquations 2.3 et 2.5 $3mR+S+T+P > m(T+P+S)+3R$ se réduit en $m>1$, donc si $m>1$, on n'a alors pas $s >_t^{p_0+2}$.

Donc si $m>1$ on peut supposer que s va jouer D contre tft au coup p_0+2.

(b) s **joue** D **contre** tft **au coup** p_0+2

	1	2	...	p_0-1	p_0	p_0+1	p_0+2
s	C	C	...	C	D	D	D
tft	C	C	...	C	C	D	D

	1	2	...	p_0-1	p_0	p_0+1	p_0+2
s	C	C	...	C	X	Y	Z
s	C	C	...	C	X	Y	Z

$$\begin{aligned} V_{p_0+2}(s) &= V_{p_0+2}(s|\texttt{tft}) + V_{p_0+2}(s|s) \\ &\leq mP + R \\ V_{p_0+2}(\texttt{tft}) &= V_{p_0+2}(\texttt{tft}|s) + V_{p_0+2}(\texttt{tft}|\texttt{tft}) \\ &= mR + P \end{aligned}$$

Finalement pour les coups p_0, p_0+1 et p_0+2 s gagne moins de $m(T+2P)+3R$ alors que tft gagne $3mR+(S+2P)$. Or toujours grâce aux inéquations de 2.3 et 2.5 $3mR+(S+2P) > m(T+2P)+3R$ se réduit en $m > \frac{3R-S-2P}{3R-T-2P}$.

Donc si $m > \frac{3R-S-2P}{3R-T-2P}$, dans le cas du dilemme classique cela se réduit en $m>3$, tft fait un meilleur score que s, et donc il n'existe pas de stratégie s qui puisse dominer M à partir d'un p_0 donné. □

[e]-améliorabilité

Un contre-exemple de panel non [e]-améliorable découle directement du résultat précédent.

3.3.3 Conclusions

Nous venons de montrer qu'il existe des panels de stratégies infiniment améliorables, mais aussi que tous les panels de stratégies ne le sont pas. En fait ceci traduit l'idée assez naturelle que certains environnements d'agents peuvent être dominés par un agent étranger, mais qu'en général il n'est pas toujours possible de construire un comportement optimal pour un environnement donné.

Cette démonstration annonce clairement le fait que l'étude de la coopération entre agents n'est pas si simple qu'on a bien voulu le penser. Elle permet également d'affirmer que l'environnement risque de jouer un rôle très important dans les comportements coopératifs.

Dans le prochain chapitre, nous allons donc étudier de plus près, non seulement la dynamique des populations, mais aussi deux stratégies particulières dont une des particularités est de modifier leur comportement en fonction de l'environnement dans lequel elles évoluent.

Chapitre 4

Simulations

Après une très brève présentation du simulateur logiciel qui a servi de support à la thèse, nous présentons quelques résultats que les simulations ont permis d'obtenir.

Tout d'abord nous avons pu vérifier de manière assez simple la robustesse du dilemme du prisonnier. Ensuite nous avons pu étudier les dynamiques de l'évolution des populations, et nous avons obtenus des résultats à la fois nouveaux et particuliers, proches de ceux de [NS89] ou [Lin92, LN95]. Nous décrivons donc la taxonomie que nous proposons pour les différentes évolutions écologiques possibles. Enfin nous présentons l'étude de deux stratégies particulières que les simulations ont permis non seulement de découvrir, mais aussi d'étudier et de comprendre. L'une d'entre elles avait été présentie par Shubik dans [Shu70].

4.1 Un logiciel de simulation

Afin de pouvoir mener à bien les différents calculs tant de tournois que d'évolutions écologiques nous utilisons un simulateur logiciel. Ce logiciel est à la base un simulateur qui a été créé afin de pouvoir gérer la compétition de stratégies informatiques pour le dilemme itéré du prisonnier organisée par Delahaye et Mathieu dans la revue Pour La Science.

Depuis il a été modifié un grand nombre de fois, subissant diverses optimisations ou modifications fondamentales de manière à être plus dynamique, et de ne pas devoir être limité statiquement à un nombre fixe de stratégies.

Au final, plusieurs versions du simulateur sont disponibles. Nous en présentons rapidement deux.

Le premier simulateur est réalisé en langage C, et se présente avant tout sous la forme d'une librairie de fonctions et objets permettant de représenter aisément les stratégies, correspondant aux comportements d'agents, les parties, les tournois ou encore les évolutions. Ensuite un simple programme permettant de choisir les stratégies à simuler, ainsi que les paramètres de simulation, et le type de simulation utilise cette librairie. La fonction première du simulateur est avant tout d'éxécuter de grandes simulations utilisant le dilemme du prisonnier. Tous les types de simulations nécessitant le calcul d'un tournoi ou d'une évolution écologique sont envisageables.

Un très grand nombre de recherches de stratégies par algorithmes génétiques ont par exemple été menées. Peu de résultats réellement concluants ayant été trouvés nous n'en faisons pas l'exposé dans la thèse. Il semble que les algorithmes génétiques ne soient pas bien adaptés à la recherche de bonnes stratégies dans le cadre général qui nous intéresse.

Toutes les simulations faites avec ce logiciel ont la possibilité d'être sauvegardées afin de pouvoir être réutilisées. Ceci est d'autant plus utile, que, comme nous l'avons vu dans le chapitre précédent, toutes les évolutions de stratégies sont basées sur le calcul de la matrice des gains impliquant toutes les stratégies à étudier. Le calcul de cette matrice est d'autant plus long que le nombre de stratégies est important. Or, devoir recalculer ces matrices pour chaque étude rendrait celle-ci sinon impratiquable du moins pénible, si, comme nous le faisons dans le chapitre suivant, un très grand nombre de stratégies sont utilisées. Pour donner un ordre d'idée, le calcul d'une matrice de gains d'un peu plus

de 1 000 stratégies prend aujourd'hui environ 10 minutes sur les stations de travail que nous avons utilisées[24]. Au début de la thèse pour le même calcul le temps demandé était de largement plus d'une heure. Cette accélération est certes due aux diverses modifications du simulateur, mais aussi et surtout, à l'augmentation de la puissance de calcul des stations de travail disponibles. La fréquence de cadence des processeurs, par exemple, a été multipliée par plus de 6 pendant les trois années qu'a duré la préparation de la thèse.

Cependant certaines limites ne sont encore pas franchissables, notamment en ce qui concerne le stockage en mémoire des données d'évolutions longues impliquant un trop grand nombre de stratégies. Aujourd'hui pour de simples limites techniques nous ne pouvons dépasser les simulations de plus de 8 000 stratégies avec ce simulateur. Il faut cependant comparer ce nombre avec les 63 stratégies des simulations d'Axelrod en 1984, [Axe84], les 100 stratégies des simulations de Delahaye et Mathieu[25] en 1992 et 1993, [DM92, DM93], ou les 32 stratégies des simulations de Meuleau et Lattaud en 1996, [ML96].

On peut finalement noter que lors des simulations, chaque stratégie utilisant la fonction de genération aléatoire est jouée un nombre `REPEAT` de fois. Ensuite la moyenne de ses `REPEAT` scores obtenus pour ses matchs, lui est attribué afin d'aplanir les effets indésirables de l'utilisation des générateurs aléatoires logiciels jamais parfaits.

C'est donc cette version du logiciel qui a été utilisée pour toutes les simulations présentées dans la thèse.

Le second simulateur est une version dont le but n'est plus l'exploration de grands espaces de stratégies mais plutôt de création de stratégies, et de tests rapides. Il sert également à faire les démonstrations des comportements des stratégies de manière visuelle.

Les deux simulateurs utilisent a priori les mêmes algorithmes de simulations, tant pour le calcul des parties, des tournois que des évolutions écologiques. Ces algorithmes sont disponibles en annexe B. Les seules différences sont le niveau d'optimisation, le langage utilisé, et l'interface avec l'utilisateur.

Le premier ne peut être manipulé que par la ligne de commande d'appel, ou la mise au point de programmes utilisant ses fonctionalités, alors que le second, écrit en JAVA, est piloté entièrement à la souris grâce à une interface utilisateur graphique. La totalité de son comportement est modifiable simplement, des paramètres de la matrice du jeu de base, jusqu'à la définition du comportement des stratégies.

Les figures 4.1 et 4.2 illustrent quelques-unes des fenêtres de l'interface de cette version du simulateur.

Nous terminons cette rapide présentation en soulignant le fait que toutes les expériences faites dans la thèse sont reproductibles aisément, et que les deux logiciels sont distribués librement sur le site web du projet PRISON : `http://www.lifl.fr/IPD`. Ce site, mis au point au cours de la thèse, est d'ailleurs utilisé pour diffuser tous les nouveaux résultats et autres informations concernant l'étude de la coopération entre agents.

4.2 Robustesse du dilemme du prisonnier

Avant de commencer l'étude de stratégies, de l'espace des stratégies, et toute autre simulation, nous avons mis au point une technique de mesure de la robustesse du jeu, pour s'assurer de la pertinence du travail et de son efficacité. Cette technique utilise des sous-ensembles de stratégies.

Un certain nombre de stratégies se retrouvent dans les différents travaux sur le dilemme. Nous avons donc étudié et mis au point un moyen de vérifier que le dilemme est un jeu dont les résultats restent stables quelque soit l'environnement de stratégies qu'il utilise.

24. Il s'agit aujourd'hui principalement de machines à base de processeur Ultra Sparc cadencé à 333 Mhz, ou de Pentium II cadencé à 450 Mhz.

25. Une évolution de 1024 stratégies avait été calculée à cette époque.

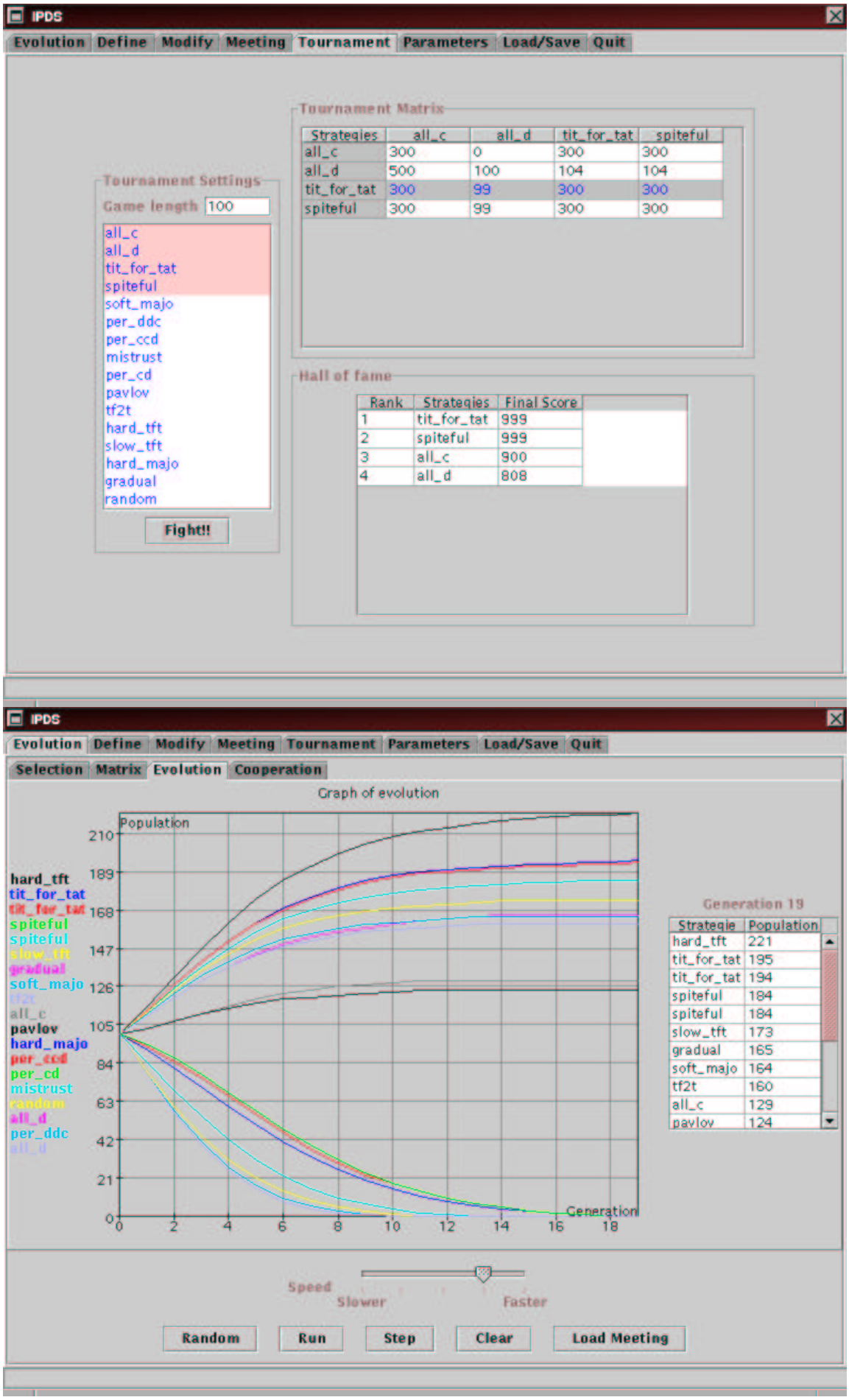


FIG. 4.1 – *Quelques vues du simulateur JAVA - Simulations.*

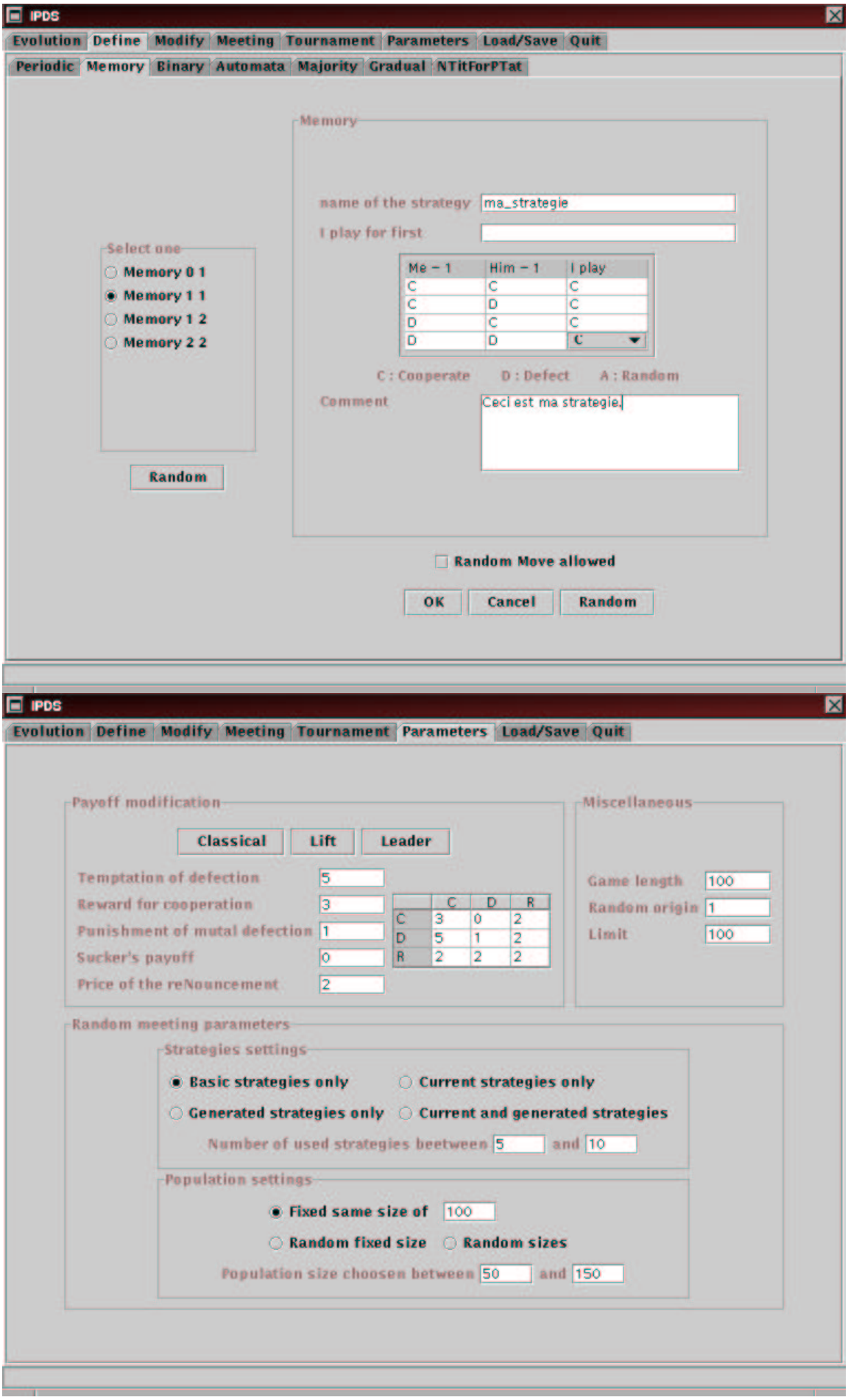


FIG. 4.2 – *Quelques vues du simulateur JAVA - Paramétrage.*

4.2.1 La méthode

Problème

La méthode utilisée est relativement simple, elle est basée sur l'idée que pour que le jeu soit intéressant à étudier, il faut que les stratégies soient stables quel que soit leur environnement. En effet il est, par exemple, bien clair que dans un environnement de stratégies bienveillantes et non réactives, la stratégie `all_d`, est une bonne stratégie, mais elle est nettement moins forte dans un environnement de stratégies réactives. La question qui nous a intéressé est de savoir si le jeu offre d'autres situations où ce cas se reproduit, et surtout combien de telles situations il peut y avoir.

Si ce nombre est trop élevé, l'intérêt d'étudier le dilemme est affaibli, car cela revient à étudier un cas beaucoup trop particulier d'évolution de la coopération.

Pour cela nous avons pensé utiliser la méthode suivante pour obtenir un indicateur de la robustesse du jeu.

Début de solution

On utilise un ensemble de N stratégies, plus ou moins représentatives de l'espace de toutes les stratégies.

On fait un tournoi généralisé entre toutes ces stratégies, ce qui permet d'obtenir un classement général C_i^g de celles-ci.

Ensuite on crée un sous-ensemble de P stratégies sur lequel on réeffectue un tournoi, ce qui permet d'obtenir un nouveau classement. Chaque stratégie est attendue à une certaine position C_i^t dans ce classement en fonction de son classement général. Si la stratégie et le jeu sont complètement stables, le classement attendu C_i^t, et le classement réel C_i^r sur ce sous-ensemble devraient coincider. Ce n'est évidemment pas le cas.

On fait alors la somme des valeurs absolues de ces différences pour chaque stratégie du sous-ensemble, ce qui permet de savoir de combien de positions le jeu a fait évoluer les stratégies.

L'idéal serait alors, pour une taille de sous-ensembles donnée, de faire ce calcul sur tous les sous-ensembles possibles, ce qui n'est malheureusement pas envisageable, car on se trouverait dans ce cas devant une explosion combinatoire.

Par exemple, pour une population de 50 stratégies dans le tournoi de départ, et une taille de sous-ensembles de 25, on se retrouve devant C_{50}^{25} sous-ensembles possibles, ce qui représente un nombre de l'ordre de 10^{14}.

La solution adoptée est donc de générer aléatoirement M sous-tournois.

Un fois ces M sous-tournois effectués, on fait la moyenne des valeurs obtenues, ce qui permet d'obtenir une mesure de la robustesse du jeu

$$R_{(P/N),M,L} = \frac{\sum_{i=1}^{i=M} \left(\sum_{j=1}^{j=P} \left| C_{i,j}^t - C_{i,j}^r \right| \right)}{M}$$

avec P le cardinal des sous-ensembles, N le nombre total de stratégies, M le nombre de mesures, L la longueur de chaque partie (nombre de coups dans une partie), et $C_{i,j}^t$, $C_{i,j}^r$ respectivement le classement attendu de la j$^{\text{e}}$ stratégie, dans le i$^{\text{e}}$ sous-ensemble, et le classement réel de cette même stratégie, dans ce même sous-ensemble.

Cette mesure de robustesse a un sens. Elle signifie qu'en moyenne une stratégie ne va pas changer son classement de plus de $R_{(P/N),M,L}$ positions.

4.2.2 Les résultats

Des évaluations ont été menées sur une population complète de 50 stratégies.

Pour ces évaluations, nous avons fait varier le nombre de coups par partie, ainsi que la taille des sous-ensembles. Le nombre de mesures étant lui fixé à $M = 1000$. Enfin ces mesures ont été faites pour deux jeux différents de scores.

Les résultats visibles sur les figures 4.3 et 4.4, mettent clairement en évidence que le jeu est assez robuste, puisqu'en moyenne, il ne modifie les positions des stratégies qu'au plus de 2,5 positions. Plus le nombre de stratégies par mesure est proche des extrémités, c'est-à-dire des nombres maximum et minimum de stratégies disponibles, plus le jeu semble robuste. Ceci est représenté par la forme en arc de cercle de chaque courbe correspondant à un choix particulier de longueur de partie.

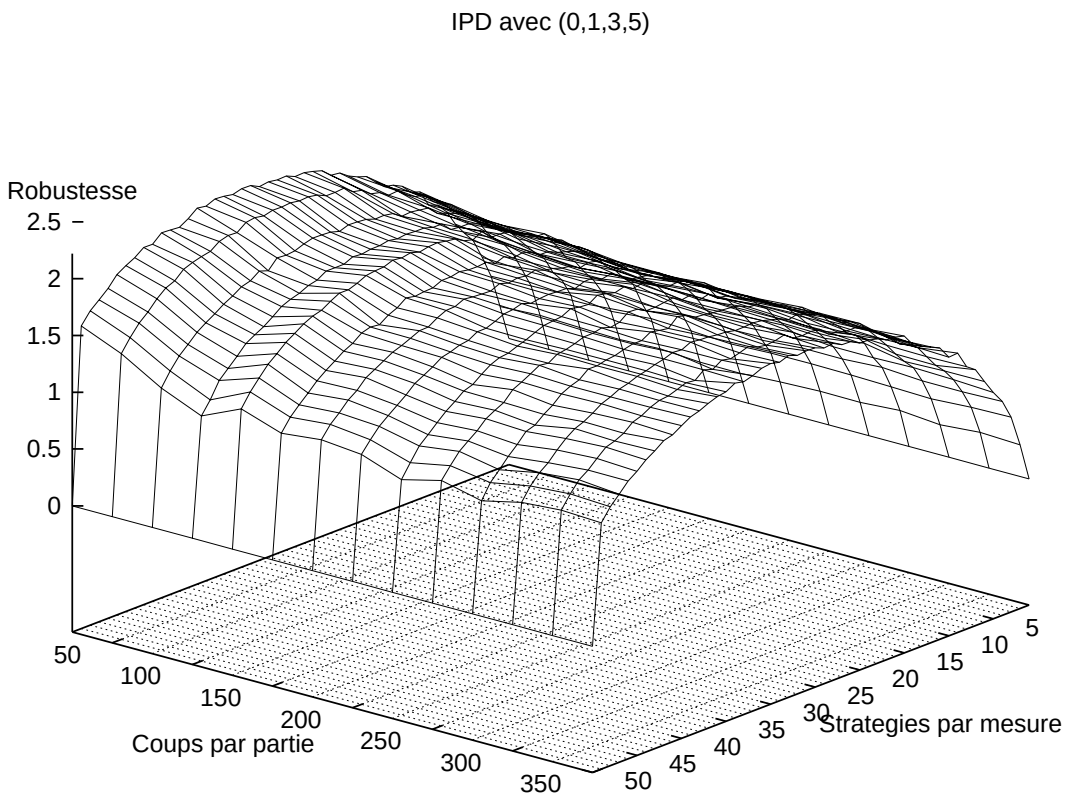


FIG. 4.3 – *Robustesse du Dilemme Itéré du Prisonnier Classique (0,1,3,5)*

De plus on peut s'apercevoir que plus le nombre de coups dans une partie est grand, plus le jeu est robuste.

Ces résultats bien qu'attendus sont importants, et inédits, et permettent de valider la poursuite des études des stratégies.

4.3 Dynamique de population avec peu de stratégies

Avant de rechercher des stratégies particulières, nous nous sommes interessés aux dynamiques que les évolutions écologiques peuvent impliquer. Afin de bien comprendre ce qui pourrait se passer dans des cas complexes nous avons étudié les cas les plus simples.

Nous avons donc cherché systématiquement parmi les évolutions impliquant trois stratégies pures des dynamiques sortant de l'ordinaire. Dans la plupart des cas, plus de 99% des milliers de cas que nous avons étudiés, les évolutions sont monotones, ce qui signifie que les courbes d'évolutions des populations croissent ou décroissent simplement.

Dans certaines autres rares situations, des oscillations complexes ont pu être observées. Il est possible, par exemple, d'obtenir des mouvements oscillatoires, croissants, décroissants, ou infinis. Il semble de plus que ces dynamiques soient en quelque sorte au *bord du chaos*, leur existence étant

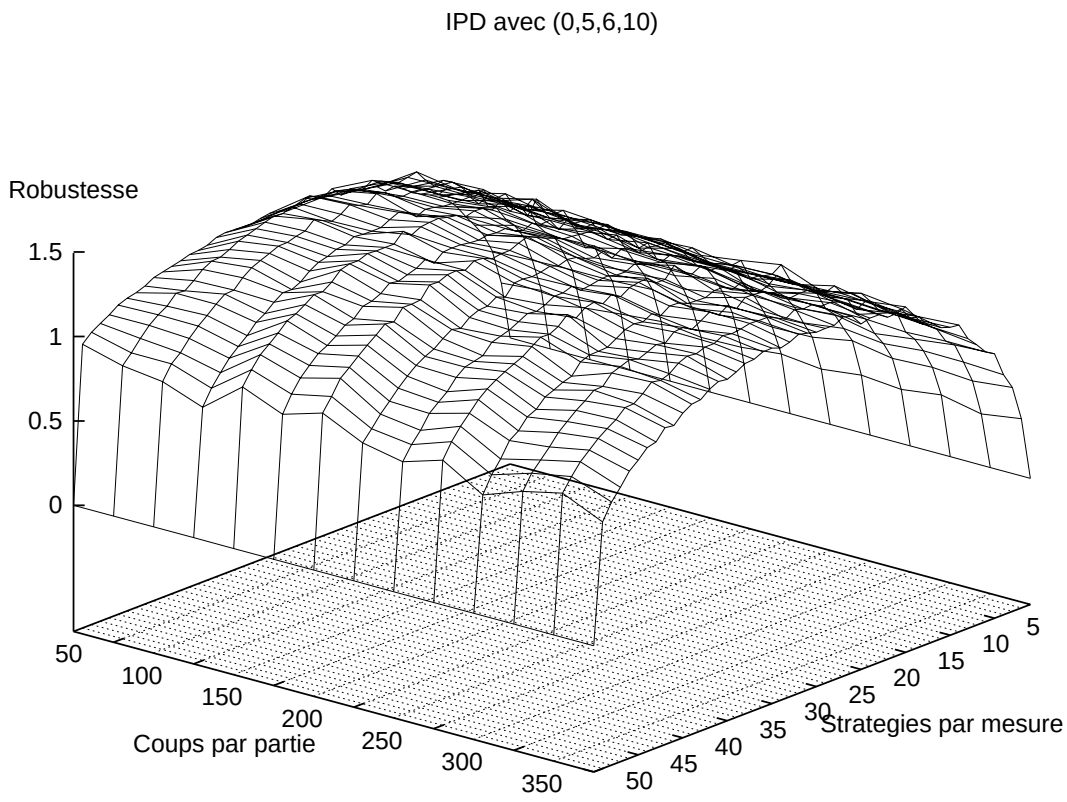


FIG. 4.4 – *Robustesse du Dilemme Itéré du Prisonnier (0,5,6,10)*

fortement liées aux conditions initiales des expériences.

Nous proposons donc une classification des dynamiques de population en 5 classes, que nous décrivons maintenant.

On doit cependant noter que de tels résultats avaient déjà été atteints dans des situations d'évolutions différentes.

Plus précisément NOWAK et SIGMUND ont mis en évidence la non stabilité à l'évolution, et en particulier la présence d'oscillations dans l'évolution des populations de trois stratégies mixtes dans [NS89]. La différence fondamentale est que ces travaux s'inscrivent dans un cadre distinct du nôtre, puisque les stratégies étudiées ne sont pas pures, et que les modèles d'évolutions ne sont pas discrets. D'ailleurs, NOWAK et SIGMUND travaillent essentiellement dans ce cadre très dissemblant du nôtre, [Now90, NS90].

NACHBAR, quant à lui, présente aussi des résultats formels et expérimentaux, exhibant de telles oscillations, [Nac92], mais une fois de plus dans un cadre différent. Le dilemme utilisé est un dilemme à horizon fini, et le modèle d'évolution utilisé est basé sur des populations infinies. Ses résultats n'en restent pas moins très intéressants, d'autant que certaines évolutions exhibées sont vraiment impressionantes.

Enfin LINDGREN, entre autres dans [Lin92] et [LN95], étudie de tels comportements sur des populations de stratégies à mémoire limitée, mais dans un environnement où le bruit, non seulement sur la communication, mais aussi sur le comportement des individus peut intervenir.

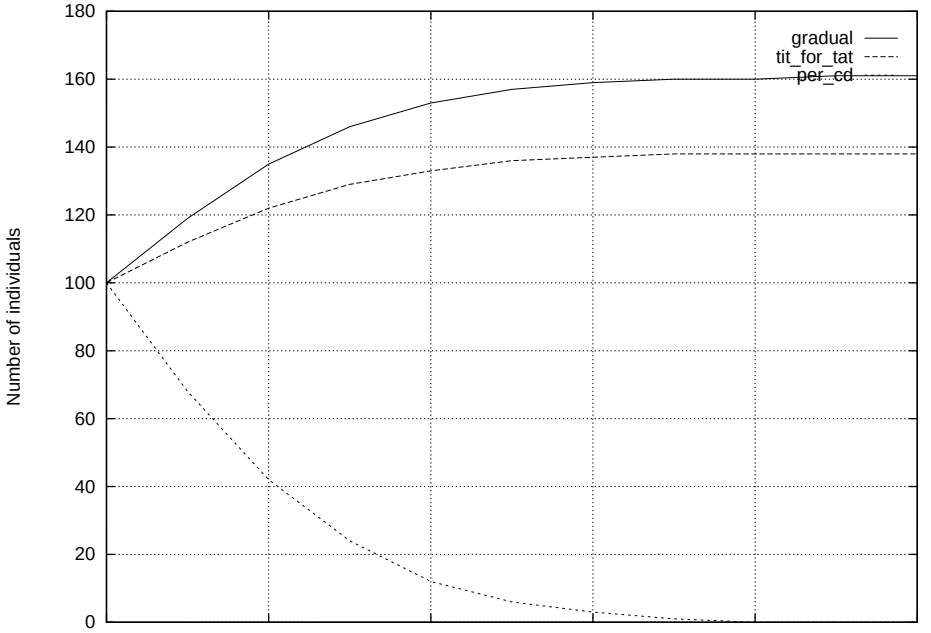
On peut rappeler que notre cadre d'étude est fort différent de ceux ci. Les cadres discrets ont été fort peu envisagés dans la littérature, alors que les dynamiques continues sont très répandues. Il est sans doute plus aisé de traiter de telles dynamiques avec des outils mathématiques, comme le calcul différentiel, cf. [HS98] pour plus de détails.

4.3.1 Dynamiques innattendues

Les travaux présentés ici ont été publiés dans [MDB99].

Convergence monotone

La première de ces 5 classes correspond à la très grande majorité des cas rencontrés (plus de 99% des cas que nous avons étudiés) et est souvent considérée comme la seule et unique façon d'imaginer l'évolution écologique. Les populations se stabilisent complètement après aucune, ou en tout cas peu de modifications (cf figure 4.5).



	tit_for_tat	gradual	per_cd	Total	Init. Pop.
tit_for_tat	3000	3000	2498	8498	100
gradual	3000	3000	2955	8955	100
per_cd	2503	615	2000	5115	100

FIG. 4.5 – *Convergence monotone*

Mouvements oscillatoires atténués

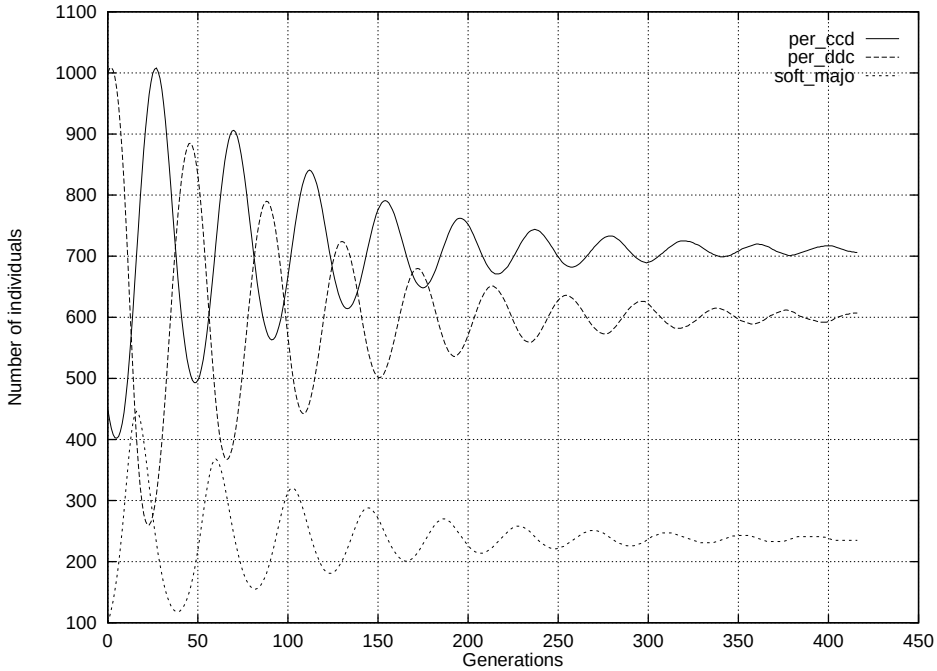
Le second cas correspond aux mouvements oscillatoires atténués. La taille des populations oscillent avec une amplitude décroissante, qui finalement permet d'obtenir une stabilité, comme dans le premier cas, mais cette fois après plusieurs retournements de situations.

La figure 4.6 illustre ce cas. Trois populations de stratégies `per_ccd`, `per_ddc` et `soft_majo` sont impliquées, et de nombreuses oscillations se produisent pendant les 100 premières générations pour graduellement se diriger vers un équilibre atteint à la génération 420.

Mouvements périodiques

Le troisième cas est celui des mouvements périodiques. Les tailles des populations après un démarrage hésitant évoluent de manière récurrente et infinie, sans jamais se stabiliser. Sur l'exemple de la figure 4.7, les tailles de populations redeviennent toujours les mêmes toutes les 37 générations. Les oscillations ne se stabilisent jamais.

Il semble que ce genre de phénomènes se produit lorsque les stratégies utilisées forment un cycle lors des tournois : `A` est meilleure que `B`, `B` est meilleure que `C` et `C` est meilleure que `A`. La nature de cette relation est peut-être une explication à la périodicité de ces mouvements.



	per_ccd	per_ddc	soft_majo	Total	Init. Pop.
per_ccd	2334	1665	3666	7665	450
per_ddc	3335	1666	671	5672	1000
soft_majo	2001	2331	3000	7332	100

FIG. 4.6 – *Mouvements oscillatoires atténués*

De tels phénomèmes ont récemment été mis en évidence par MAYNARD SMITH, [Smi96], dans le monde du vivant et plus précisément sur des populations de lézards particuliers. Même s'il n'est pas évident que le dilemme du prisonnier puisse être mis en évidence directement dans ces travaux, il est intéressant de noter la coincidence entre, d'une part le monde réel et plus particulièrement la *vie telle qu'elle est*, et d'autre part la *vie artificielle*, de nos modèles informatiques.

Oscillations croissantes

Le quatrième cas, représente les oscillations croissantes avec ruptures. Le cas représenté sur la figure 4.8 est similaire au précédent à la différence que l'amplitude des oscillations ne cesse de croître pour finalement aboutir à la disparition complète d'une des deux populations.

Dans ce cas la rupture se fait au profit de la stratégie `per_ddc`. Ce genre d'oscillations montre que les changements violents de répartition permettent à des comportements non-coopératifs de perdurer. Le désordre n'incite pas à la coopération.

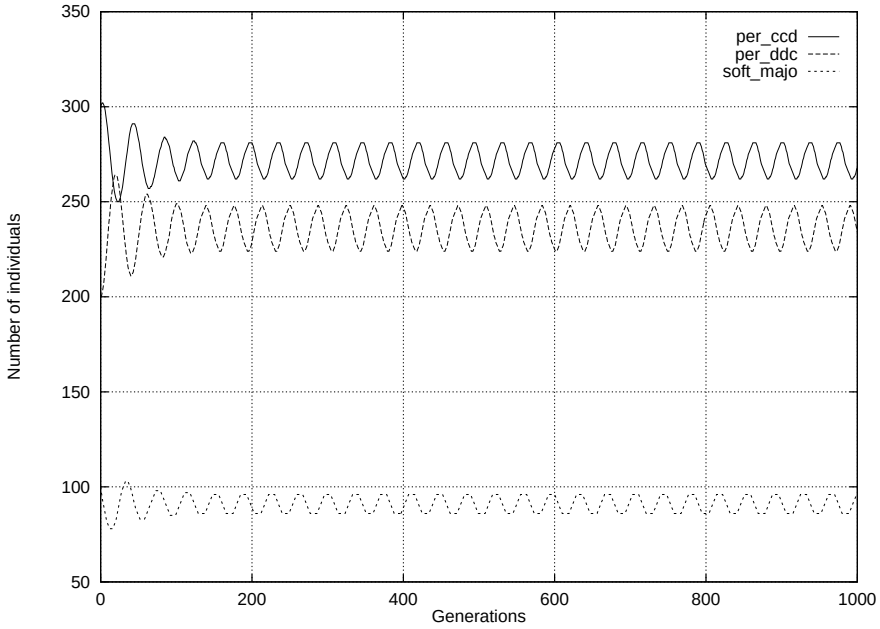
Oscillations désordonnées

Le cinquième et dernier cas correspond à toutes les oscillations que nous ne pouvons pas classer dans les catégories précédentes. Les mouvements semblent complètement désordonnés mais ne durent jamais très longtemps. Nous hésitons cependant à parler de *chaos*.

Sur la figure 4.9, après une période de 250 générations, pendant laquelle chaque stratégie frôle la disparition complète, un équilibre est finalement atteint.

4.3.2 Sensibilité aux conditions initiales

Dans le but de fixer nos idées sur l'aspect chaotique de ces évolutions, nous avons étudié la sensibilité de celles-ci aux modifications des conditions expérimentales initiales. Nous avons trouvé que



	Init. Pop.
per_ccd	300
per_ddc	200
soft_majo	100

FIG. 4.7 – *Mouvements périodiques*

	Init. Pop.
per_ccd	300
per_ddc	400
soft_majo	200

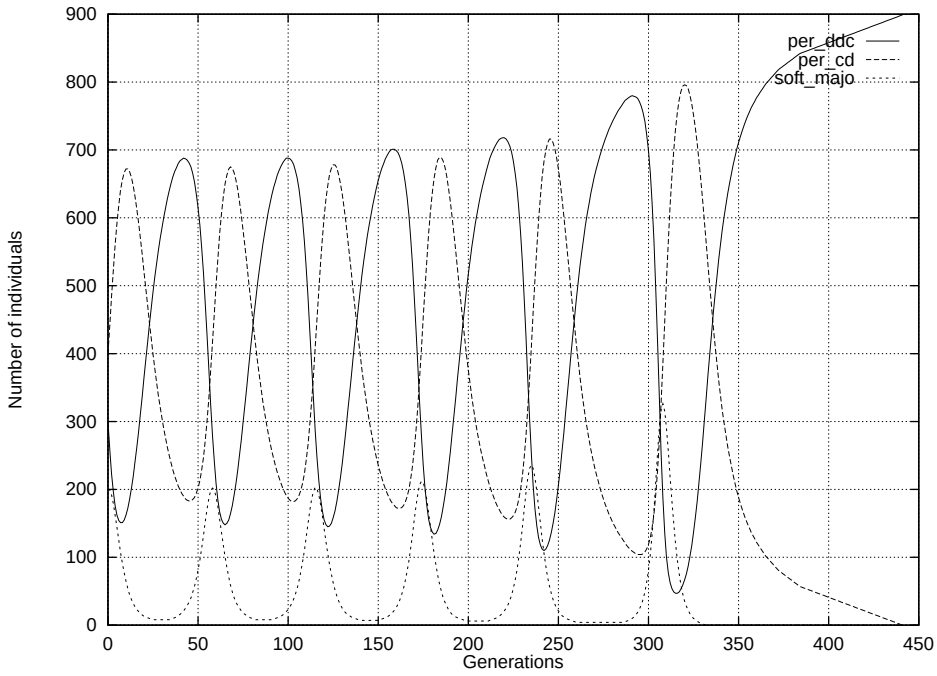
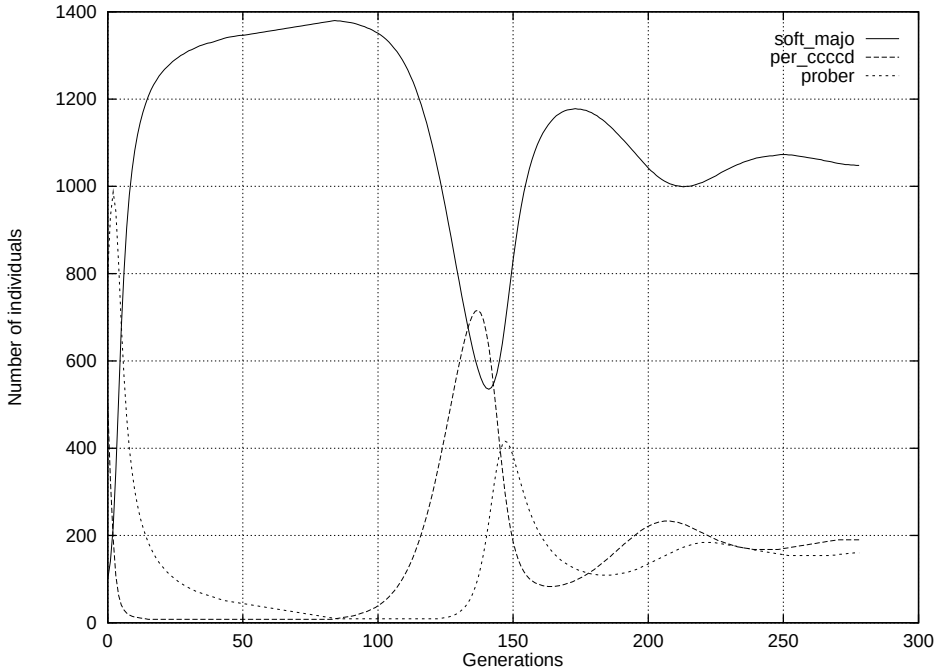


FIG. 4.8 – *Oscillations croissantes*



	soft_majo	per_ccccd	prober	Total	Init. Pop.
soft_majo	3000	2400	2999	8399	100
per_ccccd	3400	2600	206	6206	500
prober	2999	4196	1004	8199	800

FIG. 4.9 – *Oscillations désordonnées*

même d'infimes changements pouvaient entraîner des modifications de dynamique très importantes dans les phénomènes observés.

Sensibilité à la taille des populations

La transition entre une dynamique oscillatoire et une dynamique monotone peut se faire en changeant les populations de départ d'une unité.

Dans la première expérience de la figure 4.10, les paramètres utilisés sont ceux du dilemme classique, chaque partie dure 1 000 coups, au départ 300 agents utilisent `per_ccd`, 100 utilisent `soft_majo`,et 244 utilisent `per_ddc`. Les populations évoluent dans un mouvement périodique.

Si seulement un agent utilisant `per_ddc` est ajouté alors l'évolution devient monotone.

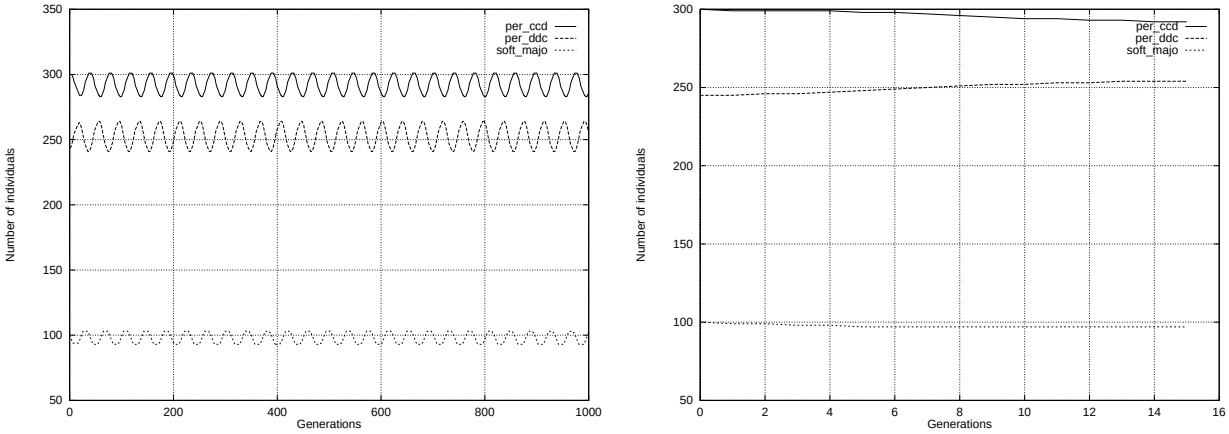


FIG. 4.10 – *Sensibilité des dynamiques aux tailles de population. Tous les paramètres sont identiques à l'exception de la taille initiale de la population* `per_ddc` *qui est de 244 à gauche et de 245 à droite.*

La modification d'une unité de la population initiale peut également modifier le vainqueur de l'évolution écologique.

Dans les expériences de la figure 4.11, les conditions sont les mêmes que précédemment à l'exception des tailles de population. Au départ, 100 agents utilisent `per_ddc`, 159 utilisent `soft_majo` et 100 utilisent `per_cd`. Le vainqueur est `per_ddc`. Si un seul agent `soft_majo` est ajouté le nouveau vainqueur est `per_cd`. On comprend assez bien comment est modifiée cette évolution en se souvenant de la manière dont agit `soft_majo`. On peut également noter que la stratégie modifiée ne gagne jamais.

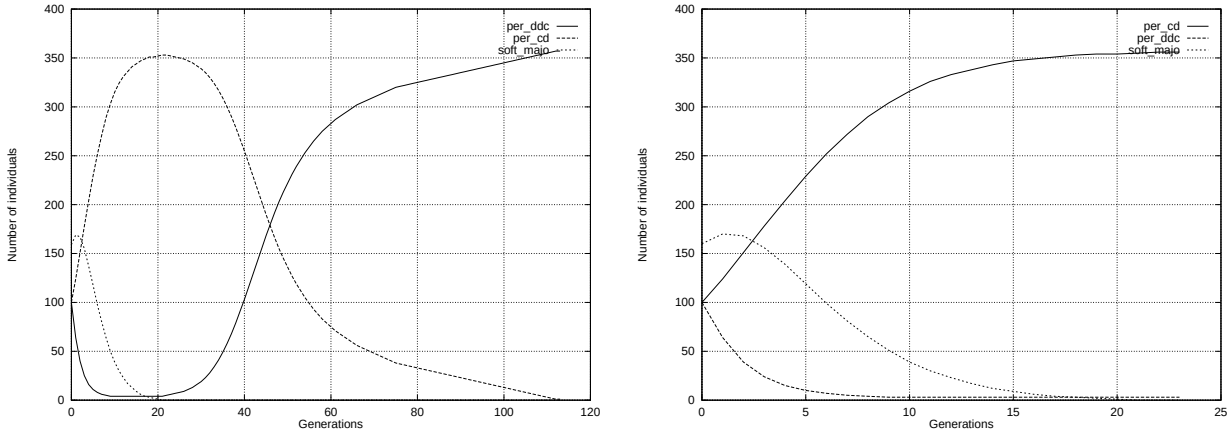


FIG. 4.11 – *Sensibilité du vainqueur aux tailles de population. Tous les paramètres sont identiques à l'exception de la taille initiale de la population* `soft_majo` *qui est de 159 à gauche et de 160 à droite.*

Sensibilité à la longueur des parties

Un changement de type de dynamique peut être observé en faisant varier la longueur des parties du dilemme, qui est fixée mais inconnue des joueurs (horizon infini).

Dans l'expérience de la figure 4.12 les paramètres du dilemme classique sont utilisés. Il y a 100 `per_ccd`, 100 `soft_majo` et 244 `per_ddc`. Quand les parties durent 7 coups, la dynamique est périodique, quand elles ne durent que 6 coups elle devient oscillatoire croissante.

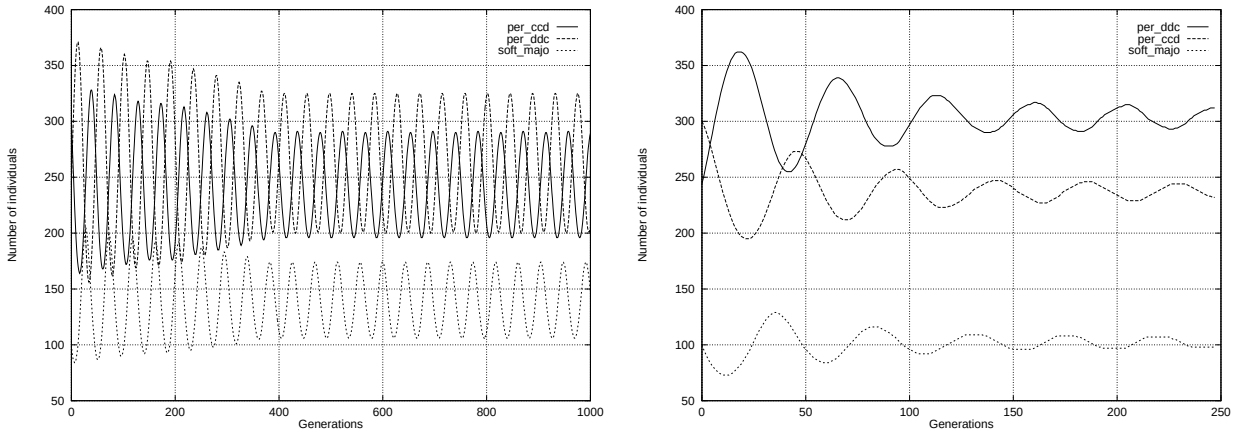


FIG. 4.12 – *Sensibilité aux longueurs de parties. Tous les paramètres sont identiques à l'exception de la durée des parties qui est de 7 coups à gauche et de 6 à droite*

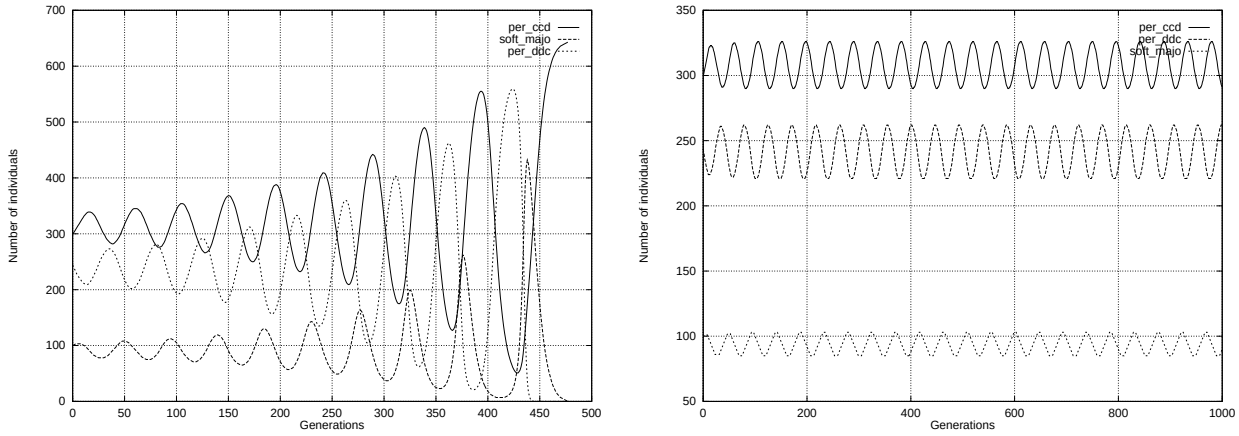


FIG. 4.13 – *Sensibilité aux paramètres du dilemme. Tous les paramètres sont identiques à l'exception de T qui vaut 4,6 à gauche, et 4,7 à droite.*

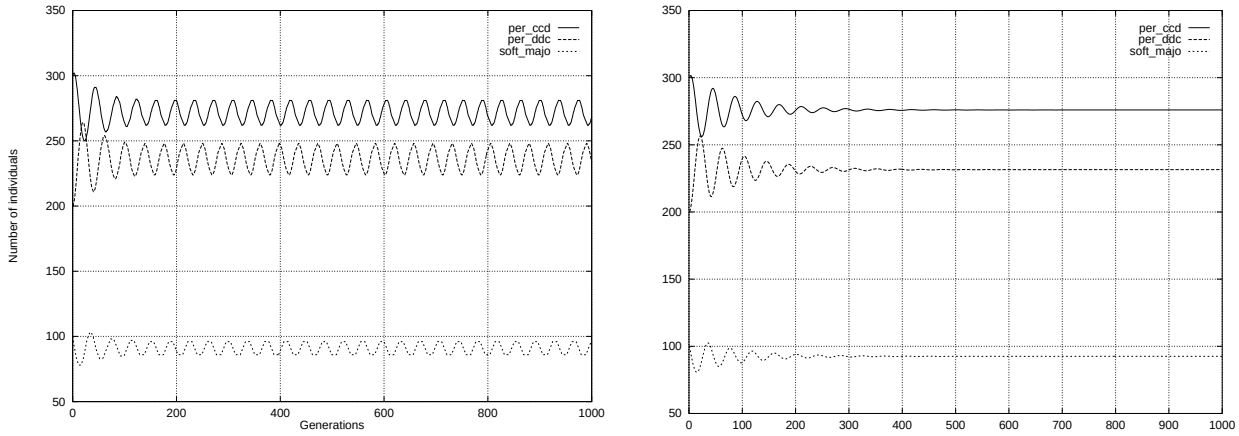


FIG. 4.14 – *Sensibilité à la normalisation. Tous les paramètres sont identiques, ormis la méthode de normalisation qui est discrète à gauche, et réelle à droite.*

Sensitibilité aux paramètres de jeu

Un changement dans la matrice de base du dilemme du prisonnier, en respectant cependant les inéquations 2.3 et 2.5, peut modifier la dynamique.

Dans les expériences de la figure 4.13 il y a 300 `per_ccd`, 100 `soft_majo` et 244 `per_ddc`. Les parties durent 1000 coups, les paramètres du dilemme classique sont utilisés en modifiant uniquement la valeur de T pour $T = 4{,}6$ et $T = 4{,}7$. Dans le premier cas on obtient un mouvement oscillatoire croissant, et dans le second un mouvement périodique.

Sensibilité à la méthode de normalisation

La dynamique peut évidemment aussi être modifiée par la méthode de répartition entre deux générations.

Dans les expériences de la figure 4.14, les paramètres du dilemme classique sont utilisés. Les parties durent 1 000 coups. Il y a 300 `per_ccd`, 100 `soft_majo` et 200 `per_ddc`. Avec la méthode d'évolution décrite dans le chapitre précédent, la normalisation de la population entre chaque génération se fait en arrondissant à l'entier le plus proche. Dans ce cas l'évolution est périodique. Si on utilise une normalisation réelle, les populations étant alors elles-mêmes réelles et plus discrètes, alors on arrive à une évolution à oscillations atténuées.

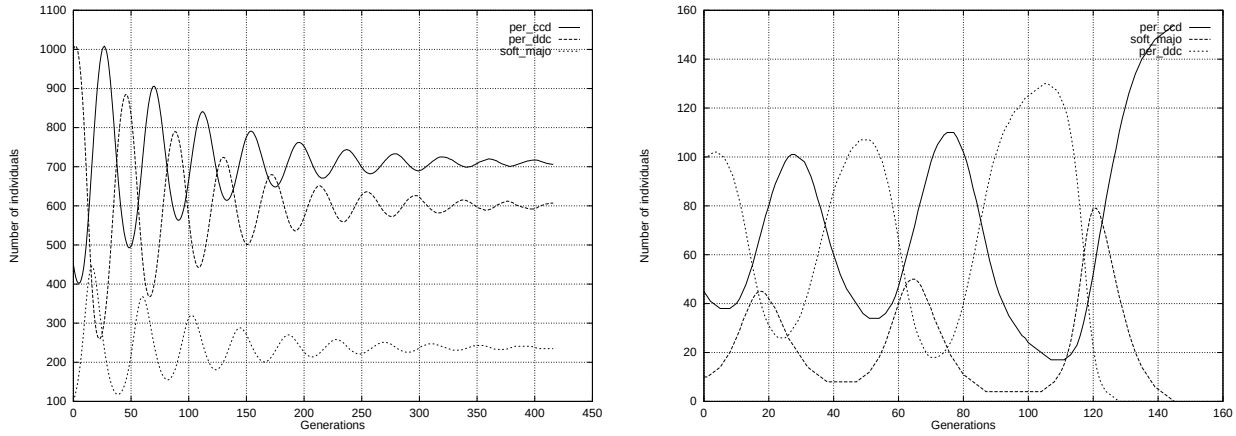


FIG. 4.15 – *Sensibilité à la méthode de répartition. Les paramètres sont identiques à l'exception des tailles de population divisées par 10 à droite.*

Cette sensibilité est confortée par le fait que la dynamique change également si les proportions des stratégies dans les populations sont modifiées par un facteur constant.

Les expériences de la figure 4.15, sont similaires à celles de la figure 4.14 à l'exception de la distribution de la population : il y a 450 `per_ccd`, 100 `soft_majo` et 1 000 `per_ddc`. La dynamique est oscillatoire croissante. Si toutes les populations sont divisées par 10, elle devient croissante.

4.3.3 Conclusion

Très rarement, les évolutions écologiques entraînent des dynamiques de population désordonnées. Nous classons ces dynamiques en 5 catégories. Dans les cas périodiques, croissants infinis ou décroissants, des stratégies méchantes sont toujours impliquées. De temps en temps ces perturbations permettent à ces stratégies de gagner l'évolution, c'est-à-dire de se répandre très largement. Il semble donc que le désordre donne plus de chances à l'agressivité, et soit donc peu favorable aux comportements coopératifs.

La sensibilité aux conditions initiales, et le fait de voir des stratégies largement non coopératives dominer certaines évolutions rend les comportements quasiment imprédictible si on se contente d'étudier les équations décrivant cette évolution.

Il est fort probable que la complexité rencontrée ici ne cesse de croître dans des situations elles-mêmes plus complexes (nombre de stratégies plus élevé, stratégie aux comportements moins simplistes).

Les relations sociales seraient alors instables par nature, du fait de dynamiques oscillatoires violentes profitant à la non-coopération.

Nous allons maintenant justement nous intéresser à deux stratégies plus complexes, qui semblent mieux se comporter dans de nombreux cas que les stratégies classiquement recommandées, et notamment qui semblent plus efficaces et robustes que `tit_for_tat`.

4.4 Une bonne stratégie : `gradual`

Pendant toute la durée de la thèse, nous avons été amenés à vérifier les résultats classiques présentés dans la littérature. Nous avons notamment testé l'efficacité des stratégies mises en avant pour leur qualité.

4.4.1 Comportement graduel

La stratégie la plus souvent mise en valeur par les différents travaux existants est sans aucun doute `tit_for_tat`. Les qualités que lui accorde AXELROD sont du même coup considérées comme absolues. Nous mettons en doute une de ses caractéristiques : la simplicité. Ce point de vue avait déjà été discuté par DELAHAYE et MATHIEU dans [DM95] à propos d'une variante du dilemme du prisonnier avec renoncement. Dans ce but, nous avons essayé d'accroître les capacités des stratégies connues en les modifiant peu à peu et en vérifiant que nos modifications amélioraient les stratégies, tant en tournoi, qu'en compétition écologique. Durant ces nombreux essais nous avons créé un grand nombre de stratégies dont la stratégie `gradual`, dont le comportement est rappelé ici :

`gradual`

Je coopère jusqu'à la première trahison de mon adversaire. Ensuite pour chaque trahison je punis mon adversaire par n trahison(s), le calme par 2 coopérations et continue à coopérer. n est le nombre de fois que mon adversaire m'a trahi.

Cette stratégie est apparue très forte par rapport aux stratégies classiques, et notamment par rapport à `tit_for_tat`.

Elle se comporte exactement comme `tit_for_tat`, excepté lorsqu'il s'agit de punir son adversaire et de se souvenir du passé. Elle coopère au premier coup et ne commence jamais à trahir avant que son adversaire ne l'ait fait. Tout comme `tit_for_tat` elle est *bienveillante*.

Après la première trahison de l'adversaire, elle se met à trahir, et change donc son comportement. Tout comme `tit_for_tat` elle est *réactive*.

Après une certaine période de punition, elle retrouve son comportement initial de coopération. Tout comme `tit_for_tat` elle est *indulgente*.

A priori elle possède les trois premières caractéristiques qu'AXELROD préconise.

Possède-t-elle la quatrième caractéristique ? Nous pensons que non. Il est évident que quantifier la complexité dans l'absolu est relativement difficile. Cependant si on la quantifie en termes de nombres d'actions effectuées par la machine de TURING correspondante, il est particulièrement clair que `tit_for_tat` est beaucoup plus simple que `gradual`. En effet `tit_for_tat` n'a besoin de connaître que la situation de jeu précédente pour déterminer son coup suivant, alors que `gradual` a besoin non seulement de cette entrée mais également d'autres informations, ne serait-ce que le nombre de fois que l'adversaire a déjà trahi. Il nous semble donc clair que le comportement de `gradual` est plus complexe à décrire que celui de `tit_for_tat`. De là vient assez naturellement qu'il est aussi plus complexe de le comprendre que celui de `tit_for_tat`.

La caractéristique que `gradual` possède en plus des trois dernières est la gradualité dans la réaction, autrement l'évolutivité de la punition. Cette caractéristique lui permet d'essayer de faire comprendre à son adversaire qu'il vaut mieux éviter d'être aggressif. Plus l'adversaire l'est plus `gradual` le sera aussi.

Les inspirations de `gradual` sont assez naturelles. La nouvelle caractéristique qu'elle exhibe peut se retrouver dans la vie quotidienne. Ce comportement est utilisé par exemple par des créanciers avec leurs débiteurs. On peut par exemple imaginer le comportement du gouvernement et des contribuables, ou des compagnies de téléphone ou d'électricité avec leurs clients.

Le comportement de `gradual` peut cependant être interprété de deux manières différentes :

- Le joueur est très offensif. Il veut forcer son adversaire à coopérer. Il lui montre donc clairement qu'il a tout intérêt à ne pas le trahir.
- Le joueur est très défensif. Il ne veut pas se faire exploiter. Il évite de plus en plus la coopération avec son adversaire, en se rapprochant de plus en plus de la solution de l'équilibre de NASH du dilemme simple.

Ces deux explications sont deux possibilités d'interprétation du jeu, comme nous avons deux approches (au moins) des relations avec les autres. Dans le premier cas on essaie d'expliquer à l'autre quel est le meilleur choix pour les deux participants, dans le second on essaie de se protéger. C'est

une sorte de choix entre l'ouverture et la fermeture de cette relation. Il est clair que dans la vie ce choix n'est pas simple. Nous pensons que `gradual`, contrairement à `tit_for_tat`, offre ce type de complexité aux joueurs.

On peut noter qu'une stratégie semblable à `gradual` est présentée par SHUBIK dans [Shu70] de cette manière[26]:

> « *Player 1: "I will play my move 1 to begin with and will continue to do so, so long as my informations shows that the other player has chosen his move 1. If my informations tells me he has used move 2, then I will use move 2 for the immediate k subsequent periods, after which I will resume using move 1. If he uses his move 2 again after I have resumed using move 1 again, then I will switch to move 2 for the $k+1$ immediately subsequent periods... and so on, increasing my retaliation by an extra period for each departure from the (1,1) steady state.* »

4.4.2 Performance

Nous avons conduit un grand nombre de simulations intégrant `gradual` et `tit_for_tat`, aussi bien en tournoi qu'en évolution écologique. Il s'est avéré qu'à chaque fois, `gradual` s'est trouvée meilleure que `tit_for_tat`.

La figure 4.16 montre le résultat du tournoi impliquant l'ensemble des stratégies définies dans le chapitre précédent, hormis `bad_bet` dont nous parlerons dans la suite de ce chapitre. La figure 4.17 montre le résultat d'une évolution écologique impliquant le même panel.

```
TOURNAMENT RANK
           1 :                    gradual = 47391
           2 :                   spiteful = 45584
           3 :                  soft_majo = 45048
           4 :                tit_for_tat = 44602
           5 :                      pavlov = 43715
           6 :            two_tit_for_tat = 42831
           7 :                   slow_tft = 42628
           8 :                       tf2t = 42495
           9 :                    per_ccd = 39938
          10 :                    easy_go = 39738
          11 :                     prober = 39101
          12 :                      all_c = 37723
          13 :                   mistrust = 37151
          14 :            worse_and_worse = 36710
          15 :                 ipd_random = 35765
          16 :                      all_d = 33814
          17 :                  hard_joss = 30723
Simulations last 3 seconds.
```

FIG. 4.16 – *Un tournoi avec* `gradual`

On voit que `gradual` surpasse nettement `tit_for_tat` aussi bien en tournoi qu'en compétition écologique. On remarque aussi sur l'évolution que `gradual` évolue à peu près de la même manière que `tit_for_tat`, à une différence d'échelle près.

Nous avons exécuté un très grand nombre de simulations en utilisant `gradual` et il nous paraît intéressant de noter que `gradual` n'a que très rarement de mauvais scores, y compris contre des adversaires agressifs. C'est peut-être de là que vient sa force. En moyenne, il semble donc que `gradual` soit *meilleure* que `tit_for_tat`.

26. La carte 1 correspond à `C` et la carte 2 à `D`.

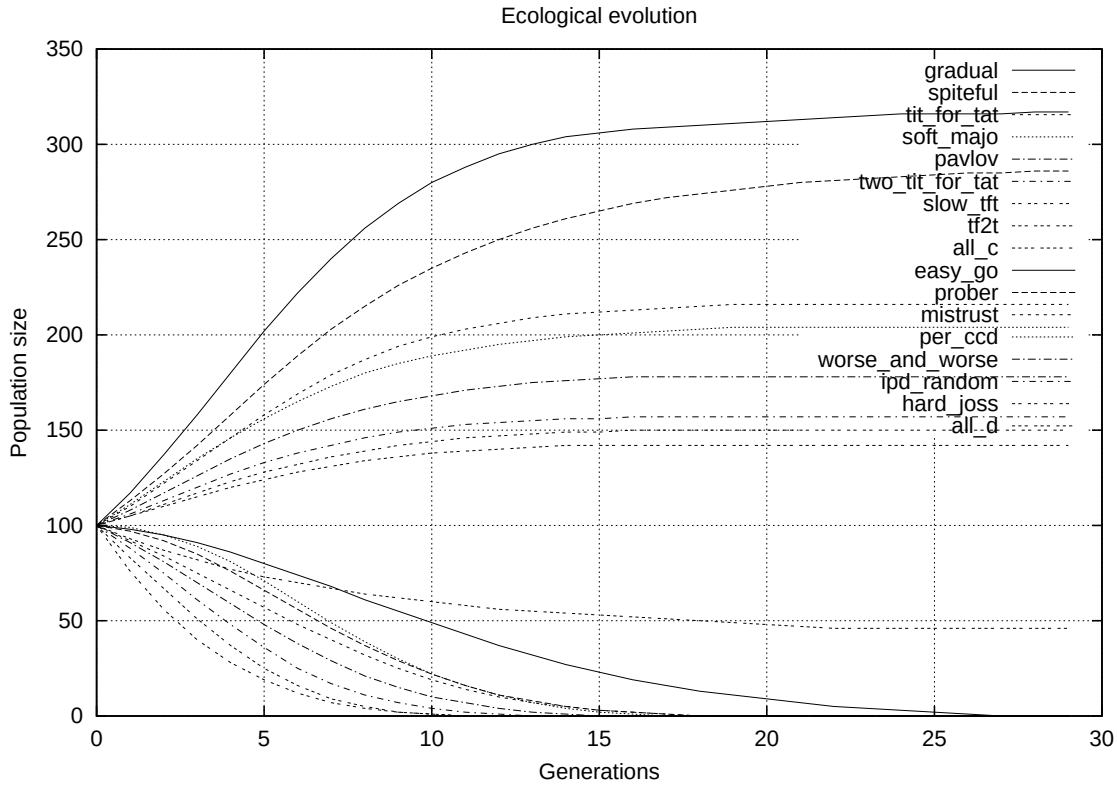


FIG. 4.17 – *Une évolution avec* `gradual`

La stratégie que nous décrivons maintenant, elle aussi, possède cette qualité de ne pas faire souvent de mauvais scores.

4.5 Une stratégie encore meilleure : `bad_bet`

Une fois de plus, comme pour le cas de `gradual`, une autre stratégie a été inspirée par les stratégies classiques ou par les résultats des différents concours organisés, notamment par DELAHAYE etMATHIEU.

Une de celles-ci est la stratégie `bad_bet` [27] dont le comportement est le suivant :

Tant que l'adversaire coopère, elle coopère. Dès qu'il a trahi une fois, elle suit ce principe :

1. Pendant 4 coups elle joue la stratégie `tit_for_tat`
2. puis pendant 4 coups elle joue `all_c`
3. puis pendant 4 coups elle joue `spiteful`
4. puis pendant 4 coups elle joue `per_ccd`
5. elle compare les scores relatifs, *i.e.* limités à la période d'essai, obtenus lors de l'essai des 4 stratégies par chacune d'entre elles, et choisit de jouer la stratégie qui a obtenu le score le plus élevé, pendant 4 coups
6. elle met à jour (simple remplacement) le score relatif obtenu par la stratégie testée
7. elle retourne au point 5

Avant d'analyser son comportement, nous présentons quelques exemples de ses performances.

27. Son nom vient d'un pari perdu fait avec Philippe MATHIEU, qui était le premier à croire en la valeur de cette stratégie.

4.5.1 Performances

Comme pour `gradual`, nous avons conduit un grand nombre de simulations intégrant `bad_bet` et `tit_for_tat`, aussi bien en tournoi qu'en évolution écologique. Il s'est aussi avéré qu'à chaque fois, `bad_bet` s'est trouvée meilleure que `tit_for_tat`.

La figure 4.18 montre le résultat du tournoi impliquant l'ensemble des stratégies définies dans le chapitre précédent, hormis `gradual`. La figure 4.19 montre le résultat d'une évolution écologique impliquant le même panel.

```
TOURNAMENT RANK
           1 :                    bad_bet = 44255
           2 :                tit_for_tat = 42161
           3 :                   spiteful = 41662
           4 :                  soft_majo = 41496
           5 :                   slow_tft = 39989
           6 :                       tf2t = 39987
           7 :            two_tit_for_tat = 39343
           8 :                     pavlov = 38831
           9 :                     prober = 35685
          10 :                   mistrust = 34686
          11 :                    per_ccd = 34398
          12 :                    easy_go = 33008
          13 :                      all_c = 32007
          14 :                  hard_joss = 29744
          15 :            worse_and_worse = 29740
          16 :                 ipd_random = 29740
          17 :                      all_d = 29740
Simulations last 3 seconds.
```

FIG. 4.18 – *Un tournoi avec* `bad_bet`

On peut voir qu'une fois de plus `tit_for_tat` est très nettement surpassée par une stratégie, ici `bad_bet` et ce, aussi bien en tournoi qu'en compétition écologique.

Afin de comparer `bad_bet` et `gradual`, nous les avons testées dans un très grand nombre d'environnements différents, et il s'est avéré que non seulement, `bad_bet` est *meilleure* que `tit_for_tat`, mais qu'elle est aussi meilleure que `gradual`. La figure 4.20 montre par exemple une évolution impliquant ces deux stratégies.

4.5.2 Comportement de `bad_bet`

D'où vient la force de `bad_bet` ? Avant de tenter de répondre à cette question, nous avons cherché à comprendre si son mécanisme de changement de stratégies était utilisé. Grâce au simulateur, nous avons pu tracer le comportement de `bad_bet` pas à pas dans un tournoi.

On peut voir sur le tableau 4.1 la trace de ce comportement dans un tournoi particulier.

Pour chacune des stratégies pour laquelle une punition a été nécessaire, on peut lire tout d'abord le numéro du coup où la punition a commencé, la date de début de punition, puis pour chaque période, le numéro de la stratégie utilisée. Les quatre premières périodes sont celles que toute stratégie `bad_bet` teste avant de commencer à faire son choix.

Pour les stratégies utilisant des choix aléatoires, seule la trace pour la première des rencontres a été conservée.

Nous avons fait de nombreuses traces de ce genre. À chaque fois, on s'aperçoit que non seulement toutes les stratégies de la gamme de punition de `bad_bet` sont utilisées, mais qu'en plus au cours d'une rencontre plusieurs stratégies différentes sont parfois utilisées pour punir un adversaire récalcitrant.

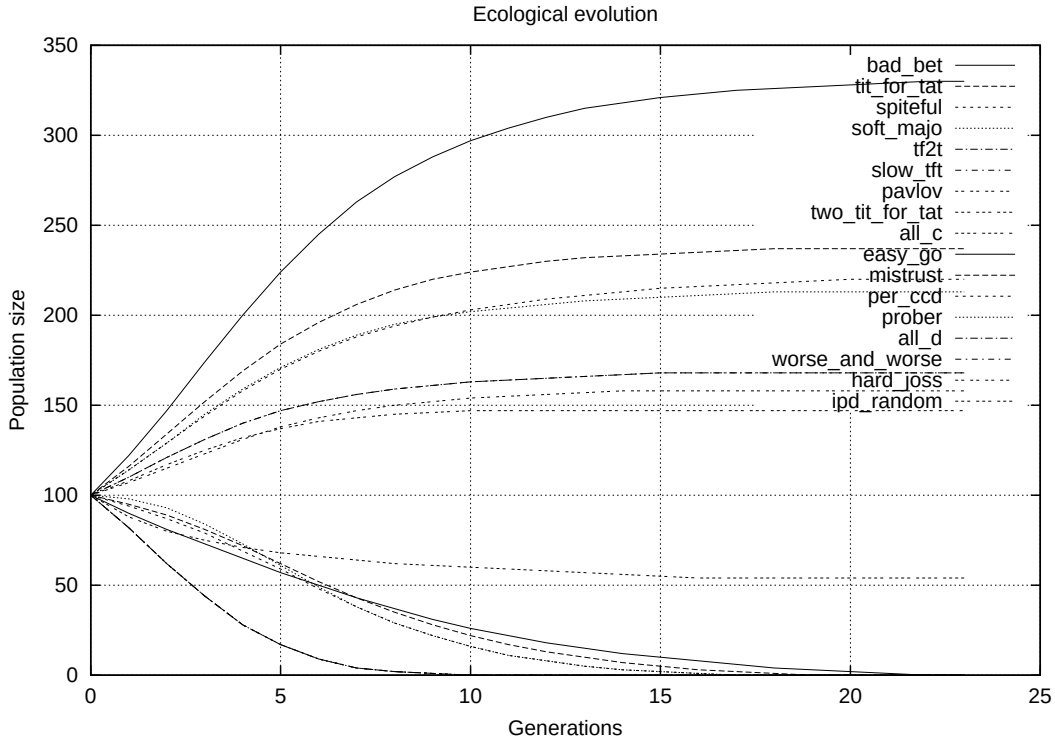


FIG. 4.19 – *Une évolution avec* `bad_bet`

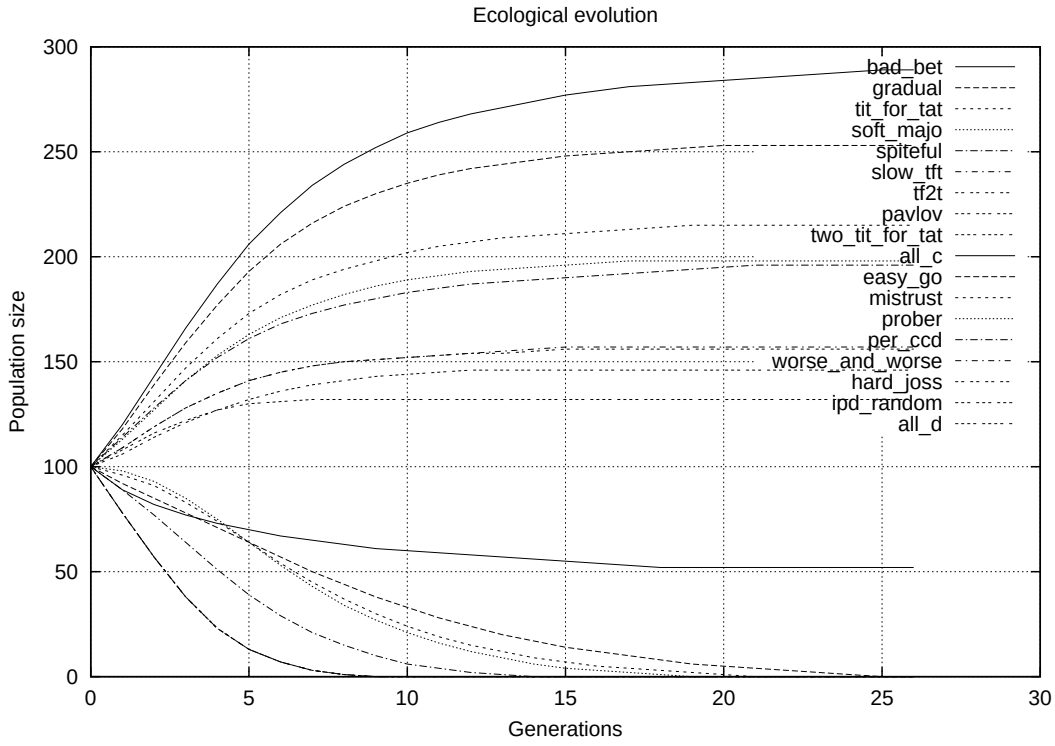


FIG. 4.20 – *Une évolution avec* `bad_bet` *et* `gradual`

L'analyse de ces traces nous permet de découvrir de nouvelles caractéristiques, qui nous semblent être des propriétés intéressantes pour la qualité d'une stratégie.

Adversaire	Date	Stratégies	
all_d	1	0123	00000000000000000000
ipd_random	2	0123	02000002222222222222
per_ddc	1	0123	00222222222222222222
per_ccd	3	0123	00222222222222222222
prober	1	0123	00000000000000000000
mistrust	1	0123	11111111111111111111
hard_majo	1	0123	33333333333333333333
hard_joss	29	0123	11111111111111
per_cd	2	0123	22222222222222222222
per_cccdcd	4	0123	20003332222222222222
prob_c_4_on_5	6	0123	00032222222222222222
per_ccccd	5	0123	02222222222222222222
prober2	1	0123	03333333333333333333
prober3	1	0123	11111111111111111111
prober4	3	0123	22211111111111111111
hard_prober	1	0123	00000000000000000000
better_and_better	1	0123	00000000000000000000
worse_and_worse	15	0123	033333333300011111
worse_and_worse2	26	0123	133302000000000
gradual_killer	1	0123	33333333333333333333
easy_go	1	0123	33333333333333333333

TAB. 4.1 – *Pistage du comportement de* bad_bet*. Les stratégies utilisées par* bad_bet *sont 0*=tit_for_tat*, 1*=all_c*, 2*=spiteful *et 3*=per_ccd

L'idée globale de bad_bet est donc celle d'une stratégie *gentille*, *réactive* mais pas forcément *indulgente*.

En fait bad_bet adapte son comportement à son adversaire. Cette adaptation est un concept que peu de stratégies utilisent. On peut par exemple dire que prober ajuste son attitude : si son adversaire se laisse faire, alors elle l'exploite sinon elle joue la sécurité en jouant comme tit_for_tat. Le problème de prober est que pour s'adapter à son adversaire elle cherche à savoir quel est son comportement : prober est agressive. bad_bet adapte sa stratégie à celle de son adversaire. Plus exactement elle ajuste son comportement quand son adversaire n'est pas coopératif. Elle punit ses adversaires différemment en fonction de la faute qu'ils ont commise. L'idée ici est que toutes les trahisons de l'adversaire ne sont pas forcément des tentatives d'exploitation. bad_bet considère qu'une trahison est une tentative de communication. Elle essaie alors différents *langages* et utilise celui qui lui coûte le moins cher. Ce choix reste cependant dynamique durant toute la longueur du reste de la partie pour le cas où l'adversaire réussit à apprendre le langage que bad_bet a choisi.

L'idée forte qui ressort de cette première étude est donc que la réaction doit être adaptative.

En se basant sur cette idée et en étudiant plus précisément les stratégies que bad_bet utilise on arrive un peu mieux à saisir les concepts que bad_bet favorise pour sa réaction.

Après une trahison elle commence par jouer tit_for_tat, c'est-à-dire que pendant quatre coups elle va jouer prudemment contre son adversaire. Ensuite elle va essayer d'utiliser la coopération pendant quatre autres coups, ici elle pardonne à son adversaire sa trahison. Ensuite elle va tenter d'être rancunière pendant les quatre coups suivants. Puis finalement elle va trahir périodiquement une fois sur trois.

Le choix de la stratégie qu'elle conserve pour la suite est souvent un choix durable, comme on peut le voir sur le tableau 4.1. Plus précisément contre per_ccccd par exemple elle joue prober, ce qui l'amène à trahir continuellement jusqu'à la fin de la partie. Or le meilleur comportement à adopter face à une stratégie périodique est bien de trahir tout le temps. Elle ne pardonne pas à son adversaire

de vouloir constamment tenter de la trahir. Contre `hard_joss` elle va finir par toujours coopérer, ce qui est aussi le comportement optimal, puisque jouer `tit_for_tat` pour essayer de récupérer les points volés par `hard_joss` entraînerait une suite de coups désastreux pour les deux. Dans ce cas, elle pardonne à son adversaire les rares trahisons qu'il commet.

En définitive on peut bien isoler les quatre types de réponse que `bad_bet` fait à un adversaire qui l'agresse :

- **La punition simple**. Dans ce cas `bad_bet` considère que son adversaire comprend bien les punitions, et qu'elles suffisent à ne pas lui faire perdre trop de points.
- **Le pardon ou l'oubli**. Dans ce cas `bad_bet` ferme les yeux sur la trahison de son adversaire, parce qu'elle considère que tenter de le convaincre lui coûterait trop cher.
- **La punition sévère**. Dans ce cas `bad_bet` comprend que l'adversaire est vraiment trop agressif ou sournois et préfère rompre toute tentative de coopération.
- **L'exploitation**. Dans ce cas `bad_bet` essaie de profiter de la naïveté de son adversaire.

Dans tous les cas, `bad_bet` accumule des qualités exhibées par d'autres stratégies, comme la gentillesse et la réactivité de `tit_for_tat`, l'évolutivité de la réaction de `gradual`, et enfin l'adaptation du comportement, et plus précisément de la réaction.

Cela conforte notre idée que la simplicité n'est pas une bonne qualité, mais qu'au contraire il semble exister une hiérarchie de stratégies aux qualités et à la complexité croissantes. Nous pensons que cette hiérarchie est infinie, et qu'il en existe sans doute plusieurs.

Pour exemple de hiérarchie, on peut citer la progression de `tit_for_tat` vers `gradual` (une première adaptation simple de la punition est utilisée) et de `gradual` vers `bad_bet` (l'adaptation est plus subtile).

Nous allons maintenant pouvoir vérifier que ces stratégies ont des bons comportements, en les testant dans de très larges environnements stratégiques. Leur robustesse sera mesurée par leur efficacité face à de nombreuses stratégies différentes. Le recours aux simulations nous sera en cela bien indispensable.

Chapitre 5

Classes complètes de stratégies

Nous exposons dans ce chapitre une méthode originale d'exploration d'espaces de stratégies : les classes complètes de stratégies. Après en avoir défini le concept et le but, nous décrivons les différentes variétés de classes utilisables avec nos simulateurs ainsi que certaines des propriétés spécifiques à celles-ci. Il est à noter que les idées utilisées ici sont en partie présente dans [Dav87, ML96]. Nous présentons ensuite les travaux effectués en décrivant les expériences menées sur certaines de ces classes et en en étudiant les résultats.

5.1 Espace complet

5.1.1 Les grands espaces de stratégies

Une des méthodes pour obtenir un grand nombre de stratégies est de demander à un grand nombre de personnes de décrire une stratégie pour le dilemme. Cette méthode, qui a fait ses preuves, a été entre autre utilisée par Axelrod, ou Delahaye et Mathieu. Dans le premier cas, la communauté scientifique gravitant autour de l'université du Michigan a été mise à contribution et dans le second tous les lecteurs de la revue généraliste d'information scientifique « Pour La Science » ont pu participer. À chaque fois cette méthode permet d'obtenir un assez grand nombre de stratégies assez diversifiées. Axelrod en a collecté une soixantaine, alors que Delahaye et Mathieu en ont collecté une centaine. Cette méthode continue à être souvent employée notamment dans les conférences sur la Vie Artificielle, comme Artificial Life V, en 1996, ou Conference on Evolutionary Computation 2000. Son aspect de compétition ludique permet d'attirer l'attention de nombreux participants.

Une autre méthode consiste à utiliser des *cobayes* humains, en laboratoire, en situation de jeu et d'observer leurs comportements. De cette observation, peuvent alors être tirées des règles plus ou moins générales. Cette méthode est très souvent utilisée en psychologie ou en sociologie. On peut se référer par exemple, à [RC65], qui est une des premières études complètes et profondes du dilemme du prisonnier. La totalité de celle-ci est basée sur des comportements humains observés dans des conditions d'expériences particulières, proches des conditions initiales de Flood et Dresher. Les règles observées ici ne sont pas sans intérêt, puisque l'auteur de cet ouvrage est aussi le père de la stratégie `tit_for_tat`, dont nous avons déjà parlé dans le chapitre 3 et qui a été mise en valeur par les expériences informatiques d'Axelrod. Plus modestement, mais plus récemment, on peut également citer les travaux d'une équipe de psychologues de l'université de Lille III initiés par [Del97].

Avec ces deux méthodes, le choix des stratégies et la nature de celles-ci ne sont pas complètement objectifs. Chaque stratégie est, plus ou moins volontairement, imprégnée de la *conscience* de son créateur.

5.1.2 Motivation

Pour faire des simulations d'évolutions écologiques conséquentes permettant de déduire des résultats suffisamment généraux, il faut disposer d'un grand nombre de stratégies.

Avoir un très grand nombre de stratégies permet de simuler des évolutions écologiques de grande ampleur et donc de tester et évaluer dans des ensembles larges et hétérogènes certaines stratégies construites sur la base de critères particuliers. Ces évaluations de grande ampleur permettent de vérifier la qualité de ces critères.

Le but avoué de la création et de l'utilisation des classes complètes est donc avant tout l'évaluation de stratégies que nous supposons être de qualité. Cette évaluation permet alors non seulement de comparer les critères de qualité des stratégies mais peut-être aussi de mieux comprendre leur fonctionnement, du fait de leur confrontation à de nombreux autres comportements différents.

Une autre motivation est de pouvoir trouver de nouvelles « *bonnes* » stratégies. En explorant systématiquement un espace de stratégies, on peut espérer trouver des stratégies dont les comportements pourront nous apporter de nouvelles idées de critères de qualité, du fait de leur bon comportement en évolution écologique. Plus cet espace sera grand, plus la recherche risque d'être fructueuse (mais également longue).

Les stratégies créées doivent l'être de la manière la plus objective possible, c'est-à-dire sans introduire trop de traits de caractères non naturels ou artifiels.

Nous proposons une idée originale et pourtant simple permettant de créer un très grand nombre de stratégies facilement et de manière la plus objective possible.

5.1.3 Concept génétique

L'idée que nous proposons est de définir un comportement général en fonction d'un jeu de paramètres, puis d'utiliser toutes les valeurs possibles de ces paramètres comme autant de définitions de stratégies différentes. Il suffit pour cela de définir une structure pouvant être décodée en un comportement pour le dilemme itéré du prisonnier, puis d'utiliser toutes les valeurs possibles pouvant remplir cette structure comme autant de stratégies différentes.

L'idée de structure n'est pas en elle-même originale. C'est l'idée de base des approches évolutionnaires ou génétiques, [Fog95]. Plus globalement c'est l'idée de base du fonctionnement des êtres humains. Chaque individu possède des gènes. Chaque gène a une fonction qui lui est propre et qui correspond à un comportement ou à une propriété telle la couleur des yeux, des cheveux etc. De manière plus simple, on peut voir cette structure comme une suite d'interrupteurs. La position de chacun d'entre eux permet de déterminer l'utilisation d'une information dans un comportement. Le comportement final dépend de la position de tous ces interrupteurs. On appelle souvent cette suite un génotype. Le comportement induit par le génotype est appelé le phénotype.

Nous proposons donc de définir un certain nombre de génotypes différents et de créer tous les individus possibles avec ce génotype, un peu comme si l'on créait tous les humains possibles.

5.2 Quelques classes de stratégies

Les définitions de classes et familles, de stratégies faites dans ce chapitre sont toujours accompagnées de leur description physique en vigueur dans le simulateur logiciel principal qui a servi de support aux expériences de la thèse.

5.2.1 Les mémoires

La définition la plus naturelle d'une stratégie pure dans le dilemme répété est la donnée des actions prévues en fonction du passé du jeu. Une stratégie pure permet pour chaque histoire du jeu de déterminer la prochaine carte que le joueur devra utiliser. Si l'on considère un dilemme à horizon infini, l'espace des stratégies est immense et impossible à parcourir complètement.

Une des limitations les plus naturelles de cet espace de stratégies consiste à considérer les stratégies ne prenant en compte qu'une partie de l'historique du jeu. Cette idée de restriction de la mémoire est d'ailleurs l'idée la plus répandue et la plus souvent utilisée, [Dav87, Axe97, ML96, vBvKP99].

Imaginons par exemple des joueurs humains. La mémoire humaine n'est pas infinie. À la centième répétition, un joueur a, de ce fait, plus de facilités pour se souvenir du dernier coup que du second ou du troisième [28].

En termes de stratégie, une telle limitation est facilement représentable. On peut par exemple considérer que plus un coup a été joué récemment, plus la stratégie a des chances de s'en souvenir et de fait de le prendre en compte dans le comportement à suivre. On peut alors supposer qu'à chaque nouveau coup une stratégie oublie le coup le plus ancien pour pouvoir se souvenir de celui qui vient d'être joué.

Une stratégie pour le dilemme itéré du prisonnier doit représenter un comportement. Dans ce but on considère qu'une stratégie doit être utilisable quel que soit la matrice de gains utilisée, tant que les inégalités 2.3 (cf. page 25) et 2.5 (cf. page 27) sont respectées. La description d'une stratégie doit donc être indépendante des valeurs d'utilités.

Pour se souvenir d'un coup, on ne peut alors se contenter du score obtenu, mais on doit se souvenir de deux cartes :

- celle jouée par l'adversaire ;
- celle jouée par le joueur.

Il est à noter que l'ensemble des stratégies ayant une mémoire des dix derniers coups est alors de l'ordre de $2^{2^{20}}$ (le calcul sera détaillé un peu plus loin), ce qui est un espace de stratégie encore immense.

Nous avons vu dans le chapitre 3 que la réactivité est un critère important dans un comportement au dilemme du prisonnier. Plus largement un comportement non réactif est peu intéressant à étudier puisque facilement exploitable. La précision et l'adaptation de la réaction dépendent en partie de la mémoire, non pas des coups du passé, mais plutôt des cartes jouées par l'adversaire dans le passé. Il n'y a pas égalité, du point de vue de l'intérêt, entre la mémoire concernant les cartes jouées dans le passé par l'adversaire et par le joueur.

En reprenant le cas de joueurs humains, il semble évident qu'un joueur mettra plus facilement en cause le comportement de son adversaire que le sien lors du choix d'une carte à jouer et se souviendra plus du comportement de son adversaire que du sien.

On peut raisonnablement penser que la mémoire dont doit disposer une stratégie ne doit pas être symétrique, c'est-à-dire qu'elle ne doit pas permettre de se souvenir de coups mais de cartes. La distribution de la mémoire pour le stockage des cartes jouées par l'adversaire et des cartes jouées par le joueur dans le passé n'est pas obligatoirement équitable.

Cette idée toute simple sera utilisée dans toutes les classes de stratégies que nous allons définir. La première de celle-ci ne se base que sur cette idée.

La famille `memory`

Les stratégies de la famille `memory` ont une vue limitée des cartes jouées dans le passé par l'un comme l'autre des joueurs. La limitation sur la taille de la mémoire est tout naturellement représentée par le nombre de cartes que la stratégie peut utiliser pour décider de son comportement. Comme nous l'avons dit plus haut, la taille de la mémoire des cartes jouées par l'adversaire et par le joueur peut être différente. Pour atteindre ce but, la famille `memory` est divisée en classe chacune étant définie par deux entiers :

- M, correspond au nombre de cartes du joueur qui peuvent être mémorisées ;
- O, correspond au nombre de cartes de l'adversaire qui peuvent être mémorisées.

28. Nous ne citons pas le premier coup à dessein. En effet, le premier coup peut avoir été mémorisé de part sa nature initiatrice et même avoir eu une importance non négligeable dans le comportement du joueur.

Les cartes mémorisées sont gérées chacune par une file, c'est-à-dire une liste FIFO. Après chaque coup les cartes jouées lors de celui-ci sont ajoutées dans la file respective, qui se met alors à jour. Au coup n, un joueur se rappelle des cartes qu'il a jouées au coup $n - \mathrm{M}$, ..., $n - 2$, $n - 1$ et des cartes que l'adversaire a jouée au coup $n - \mathrm{O}$, ..., $n - 2$, $n - 1$.

Afin que ce comportement puisse être établi, il faut donc que la mémoire soit remplie, c'est-à-dire que les files d'attentes soient pleines. Pour remplir la mémoire il faut alors avoir joué max(M,O) coups et donc définir les cartes qu'il faut jouer tant que ceci n'est pas satisfait. On appellera les max(M,O) coups l'*amorce* de la partie pour la stratégie. Dans le cas de la famille `memory`, cette période est décrite simplement en spécifiant en extension les cartes à jouer à chacun des coups de l'amorce de la partie par la stratégie.

Au final, pour définir une stratégie de cette classe il faut donc spécifier :

- la valeur de M,
- la valeur de O,
- l'amorce de la partie,
- la définition du comportement en fonction des historiques possibles.

Le nom d'une stratégie de cette classe sert de support à la définition de son comportement, comme c'est le cas dans le simulateur.

Par convention on représente les cartes du dilemme du prisonnier par des caractères :

- `c` correspond à la coopération (`C`) ;
- `d` correspond à la trahison (`D`).

Le nom d'une stratégie est créé par concaténation :

1. du préfixe `mem` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. des cartes à jouer pour chacun des coups de l'amorce dans l'ordre croissant des coups, *i.e* du coup 1 au coup max(M,O) ;
8. de la liste ordonnée des cartes à jouer pour chacune des configurations de la mémoire possible.
   L'ordre des configurations est déterminé en considérant que les cartes du joueurs sont placées avant celle de l'adversaire et que les cartes anciennes sont placées avant les cartes récentes.
   L'état de la mémoire à un moment donné est donc représenté par une chaîne de M+O caractères. Il y a donc exactement $2^{\mathrm{M+O}}$ configurations différentes de la mémoire.
   Les cartes à jouer sont indiquées pour chacune de ces configurations possibles. Les configurations sont classées dans l'ordre lexicographique de leur représentation.

Une seule et même chaîne de caractères est utilisée pour les points 7 et 8. Cette chaîne est donc composée de $(\max(\mathrm{M,O}) + 2^{\mathrm{M+O}})$ caractères. On appellera l'ensemble composé de cette chaîne de M et de O le *génotype* de la stratégie. On confondra souvent le nom de la stratégie et son génotype dans les familles décrites ici.

La stratégie `mem_1_2_cccccdcccd`, par exemple, est une stratégie de la famille `memory`. Elle appartient à la classe des stratégies ne se souvenant que de la dernière carte jouée par le joueur et des deux dernières carte jouées par son adversaire.

Les différentes configurations possibles pour sa mémoire sont donc :

1. il a joué `C` et son adversaire `C` suivi de `C` (on note `ccc`)
2. il a joué `C` et son adversaire `C` suivi de `D` (on note `ccd`)
3. il a joué `C` et son adversaire `D` suivi de `C` (on note `cdc`)
4. il a joué `C` et son adversaire `D` suivi de `D` (on note `cdd`)

5. il a joué D et son adversaire C suivi de C (on note dcc)
6. il a joué D et son adversaire C suivi de D (on note dcd)
7. il a joué D et son adversaire D suivi de C (on note ddc)
8. il a joué D et son adversaire D suivi de D (on note ddd)

Les cartes a jouer pour chacune des configurations précédentes sont fournies dans le même ordre que ces configurations.

La stratégie mem_1_2_cccccdcccd peut être interprétée de la manière suivante :

mem_1_2_cccccdcccd

1. *Au premier coup je joue la carte* C,
2. *au second coup je joue la carte* C,
3. *ensuite :*
   - *si j'ai joué* C, *et que mon adversaire a joué* C, *puis* C, *alors je joue* C
   - *si j'ai joué* C, *et que mon adversaire a joué* C, *puis* D, *alors je joue* C
   - *si j'ai joué* C, *et que mon adversaire a joué* D, *puis* C, *alors je joue* C
   - *si j'ai joué* C, *et que mon adversaire a joué* D, *puis* D, *alors je joue* D
   - *si j'ai joué* D, *et que mon adversaire a joué* C, *puis* C, *alors je joue* C
   - *si j'ai joué* D, *et que mon adversaire a joué* C, *puis* D, *alors je joue* C
   - *si j'ai joué* D, *et que mon adversaire a joué* D, *puis* C, *alors je joue* C
   - *si j'ai joué* D, *et que mon adversaire a joué* D, *puis* D, *alors je joue* D

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :

C | C || C C C D C C C D

Lors des deux premiers coups, cette stratégie indique au joueur qui l'utilise qu'il doit coopérer. Elle lui indique qu'ensuite il ne doit trahir, c'est-à-dire jouer D, seulement lorsque l'adversaire a trahi lors des deux derniers coups, c'est-à-dire quand sa mémoire est dans l'état cdd ou bien ddd.

On reconnaît d'ailleurs le comportement de la stratégie mem_1_2_cccccdcccd comme étant le même que celui de la stratégie tf2t, ce qui revient à dire que ces deux stratégies sont identiques. Malgré la simplicité apparente de la définition des stratégies de la famille memory, on s'aperçoit que bon nombre des stratégies connues, ou du moins rencontrées au cours de la thèse sont descriptibles dans cette classe. On peut pour l'exemple exhiber tit_for_tat qui n'est autre que mem_0_1_ccd et mem_1_1_ccdcd, ou bien spiteful qui n'est autre que mem_1_1_ccddd.

Cela permet de remarquer que non seulement le choix de définition de cette classe est naturel et judicieux, dans la mesure où de nombreuses stratégies classiques peuvent y être décrites aisément, mais aussi, pour les mêmes raisons, qu'il n'est pas aussi limitatif que l'on pourrait le croire.

Chaque classe de stratégies de la famille memory permet donc de définir $\mathcal{N}_{\texttt{mem}}(\mathrm{M},\mathrm{O})$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi dans l'ensemble {c,d} :

$$\mathcal{N}_{\texttt{mem}}(\mathrm{M},\mathrm{O}) = 2^{\left(\max(\mathrm{M},\mathrm{O})+2^{(\mathrm{M}+\mathrm{O})}\right)} \tag{5.1}$$

La famille memory_with_dynamic_start

Dans les stratégies de la famille memory, les cartes jouées lors des coups de l'amorce de la partie sont fixes. Dans ce démarrage du jeu, il est donc impossible au joueur utilisant une telle stratégie d'être réactif. Les stratégies de la classe memory_with_dynamic_start corrigent en partie ce défaut.

La description des stratégies de cette classe est identique à celle de la classe memory, seule la donnée des cartes à jouer pendant l'amorce diffère. Elle est maintenant dynamique. Les stratégies

de cette famille vont pouvoir réagir au comportement de l'adversaire y compris pendant la durée de l'amorce de la partie.

Les stratégies de cette famille, comme celles de toutes les familles à mémoires avec amorce dynamique, ne sont donc intéressantes à étudier et à utiliser que lorsque max(M,O) > 1. Dans le cas contraire, les classes complètes `memory` et `memory_with_dynamic_start`, par exemple, sont strictement équivalentes.

Le nom d'une stratégie est créé par concaténation :

1. du préfixe `memd` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. de la carte à jouer lors du premier coup ;
8. de la liste ordonnée des cartes à jouer pour chacun des coups suivant de l'amorce dans l'ordre croissant des coups, *i.e* du coup 2 au coup max(M,O) et en fonction de la carte jouée par l'adversaire au coup précédent.
   Le schéma reproduit ici est en fait le même que celui qui est utilisé dans le point suivant, ou dans le point 8, page 82, pour les stratégies de la famille `memory`. Dans les coups d'amorce, la stratégie préconise les cartes à jouer en se basant sur une représentation du passé qui est simplifiée à l'extrême : pour chacun des coups d'amorce, la stratégie ne se souvient que de la carte jouée précédemment par son adversaire.
   L'ordre de description est une fois de plus l'ordre lexicographique, c'est-à-dire la carte à jouer si l'adversaire a coopéré (`c`) puis la carte à jouer si l'adversaire a trahi (`d`).
   Les cartes présentes dans le génotype au titre du comportement pendant l'amorce ne sont donc plus simplement au nombre de (max(M,O)), mais de $(1+2(\max(M,O)-1))$ soit $(2\max(M,O)-1)$.
9. des cartes à jouer pour chacune des configurations de la mémoire possible exactement comme nous l'avons décrit pour le point 8, page 82, à propos de la famille `memory`.

Une seule et même chaîne de caractères est utilisée pour les points 7, 8 et 9. Cette chaîne est donc composée de $2\max(M,O) - 1 + 2^{M+O}$ caractères.

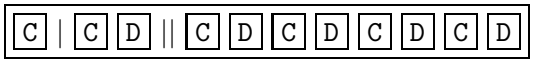
`memd_1_2_ccdcdcdcdcd`, est une stratégie de la famille `memory_with_dynamic_start`. Elle appartient à la classe des stratégies ne se souvenant que de la dernière carte jouée par le joueur, des deux dernières carte jouées par l'adversaire et ayant un démarrage dynamique.

Elle peut être interprétée comme :

`memd_1_2_ccdcdcdcdcd`

1. *Au premier coup je joue la carte* `C`,
2. *au second coup si l'adversaire a joué* `C` *je joue* `C`, *s'il a joué* `D` *alors je joue* `D`
3. *ensuite :*
   - *si j'ai joué* `C`, *et que mon adversaire a joué* `C`, *puis* `C`, *alors je joue* `C`
   - *si j'ai joué* `C`, *et que mon adversaire a joué* `C`, *puis* `D`, *alors je joue* `D`
   - *si j'ai joué* `C`, *et que mon adversaire a joué* `D`, *puis* `C`, *alors je joue* `C`
   - *si j'ai joué* `C`, *et que mon adversaire a joué* `D`, *puis* `D`, *alors je joue* `D`
   - *si j'ai joué* `D`, *et que mon adversaire a joué* `C`, *puis* `C`, *alors je joue* `C`
   - *si j'ai joué* `D`, *et que mon adversaire a joué* `C`, *puis* `D`, *alors je joue* `D`
   - *si j'ai joué* `D`, *et que mon adversaire a joué* `D`, *puis* `C`, *alors je joue* `C`
   - *si j'ai joué* `D`, *et que mon adversaire a joué* `D`, *puis* `D`, *alors je joue* `D`

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :



La remarque que nous avions faite à propos de l'expressivité de la classe `memory`, est encore valable ici. On peut par exemple identifier `memd_1_2_ccdcdcdcdcd` à `tit_for_tat`.

Chaque famille de stratégies de la classe `memory_with_dynamic_start` permet donc de définir $\mathcal{N}_{\texttt{memd}}(\mathrm{M},\mathrm{O})$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi dans l'ensemble {`c`,`d`} :

$$\mathcal{N}_{\texttt{memd}}(\mathrm{M},\mathrm{O}) = 2^{\left(2\max(\mathrm{M},\mathrm{O})-1+2^{(\mathrm{M}+\mathrm{O})}\right)} \tag{5.2}$$

La dynamicité utilisée pour l'amorce dans cette famille peut sembler ne pas être naturelle. Il peut en effet paraître plus évident de choisir une dynamicité adaptative, qui permettrait d'augmenter la qualité de la réaction au fur et à mesure de l'avancement dans l'amorce. Dans un tel cas, les cartes jouées ne dépendent pas uniquement de la dernière carte jouée par l'adversaire, mais bel et bien de toutes les cartes jouées par l'adversaire, dans la limite de O, et de toutes les cartes jouées par le joueur, dans la limite de M. Le choix arbitraire qui a été fait ici est lié à notre objectif initial : la conduite d'évolutions écologiques impliquant toutes les stratégies d'une classe. En choisissant d'utiliser une dynamicité adaptative il s'avère que le nombre de stratégies disponibles dans chaque classe devient vite beaucoup trop important pour que les simulations puissent être conduites. Le choix de cette dynamicité *relative* est donc purement expérimental.

La famille `memory_with_limited_dynamic_start`

Cette classe très artificielle a été créée dans le but de définir une classe complète particulière (pour des valeurs particulières M et O) plus large que celle accessible par la classe `memory_with_dynamic_start`. Le principe de définition de cette classe est identique au précédent, seule la dynamicité des cartes à jouer durant l'amorce est modifiée.

Cette dernière est tout simplement limitée au dernier coup de l'amorce et ne porte que sur l'avant-dernière carte jouée par l'adversaire dans cette amorce.

Le nom d'une stratégie est créé par concaténation :

1. du préfixe `memld` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. des cartes à jouer pour chacun des coups de l'amorce sauf le dernier, *i.e* du coup 1 au coup $(\max(\mathrm{M},\mathrm{O}) - 1)$.
8. de la dernière carte à jouer dans l'amorce en fonction de la carte précédemment jouée par l'adversaire.
   Le schéma est une fois de plus le même que précédemment et nous ne le répétons donc pas ici. Les cartes présentes dans le génotype au titre du comportement pendant l'amorce ne sont donc maintenant qu'au nombre de $(\max(\mathrm{M},\mathrm{O}) - 1 + 2)$ soit $(\max(\mathrm{M},\mathrm{O}) + 1)$.
9. de la liste ordonnée des cartes à jouer pour chacune des configurations de la mémoire possible exactement comme nous l'avons décrit pour le point 8, page 82, pour la famille `memory`.

Une seule et même chaîne de caractères est utilisée pour les points 7, 8 et 9. Cette chaîne est donc composée de $(\max(\mathrm{M},\mathrm{O}) + 1) + 2^{(\mathrm{M}+\mathrm{O})}$ caractères.

La stratégie `memld_0_3_ddcdccccdddd` par exemple, est une stratégie de la nouvelle classe `memory_with_limited_dynamic_start`. Elle appartient à la famille des stratégies ne se souvenant que des trois dernières cartes de son adversaire et à une amorce dynamique lors du dernier coup de celle-ci.

Elle peut être interprétée comme :

`memld_0_3_ddcdccccdddd`

1. *Au premier coup je joue la carte* [D],
2. *au second coup je joue la carte* [D],
3. *au troisième coup si l'adversaire a joué* C *je joue* [C], *s'il a joué* D *alors je joue* [D]
4. *ensuite :*
   - *si l'adversaire a joué* C, *suivi de* C, *et finalement de* C, *alors je joue* [C]
   - *si l'adversaire a joué* C, *suivi de* C, *et finalement de* D, *alors je joue* [C]
   - *si l'adversaire a joué* C, *suivi de* D, *et finalement de* C, *alors je joue* [C]
   - *si l'adversaire a joué* C, *suivi de* D, *et finalement de* D, *alors je joue* [C]
   - *si l'adversaire a joué* D, *suivi de* C, *et finalement de* C, *alors je joue* [D]
   - *si l'adversaire a joué* D, *suivi de* C, *et finalement de* D, *alors je joue* [D]
   - *si l'adversaire a joué* D, *suivi de* D, *et finalement de* C, *alors je joue* [D]
   - *si l'adversaire a joué* D, *suivi de* D, *et finalement de* D, *alors je joue* [D]

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :

[D] | [D] | [C] [D] || [C] [C] [C] [C] [D] [D] [D] [D]

Chaque classe de stratégies de la famille `memory_with_limited_dynamic_start` permet donc de définir $\mathcal{N}_{\texttt{memld}}(\mathrm{M},\mathrm{O})$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi dans l'ensemble {c,d} :

$$\mathcal{N}_{\texttt{memld}}(\mathrm{M},\mathrm{O}) = 2^{\left(\max(\mathrm{M},\mathrm{O})+1+2^{(\mathrm{M}+\mathrm{O})}\right)} \tag{5.3}$$

Il est très clair que jusqu'à présent cette famille est la plus artificielle que nous ayons décrite. Cette artificialité est toutefois toute relative puisqu'elle ne repose que sur la méthode d'initialisation de la mémoire des joueurs, le principe de base d'utilisation de cette mémoire restant quant à lui des plus naturels. C'est lui qui sera utilisé le plus fréquemment et qui donc décrit le comportement à long terme des joueurs utilisant les stratégies de cette famille.

5.2.2 Les mémoires binaires

Le concept de base des stratégies de la famille `memory` ne permet pas de prendre en compte le comportement de l'adversaire depuis le début de la partie. En effet la mémoire étant limitée en taille il est impossible pour une stratégie d'une des classes `memory` (`mem`, `memd` ou `memld`) de prendre en compte le comportement de l'adversaire de manière globale. Ces stratégies sont donc faiblement réactives, *i.e.* réactives à court terme, puisqu'elles ne réagissent que par rapport à un moment limité du passé.

Les stratégies de la famille que nous allons décrire maintenant proposent une adaptation de la structure de la classe `memory` afin de pouvoir tenter de prendre en compte le comportement global de l'adversaire depuis le début de la partie.

La classe `binary`

Les stratégies de cette famille ont basiquement les mêmes propriétés que celles de la famille `memory`, avec en plus la possibilité de différencier à chaque coup, les cas où l'adversaire a plus souvent trahi que coopéré depuis le début de la partie. Cette notion est une représentation simplifiée du comportement global de l'adversaire depuis le début de la partie.

Si l'on reprend l'exemple des joueurs humains, il semble assez naturel de considérer que les joueurs ne se souviennent certes pas avec précision de tous les coups de l'adversaire, mais que globalement ils se font une idée de leur adversaire en se souvenant de son comportement en moyenne. La mémoire des coups récents reste cependant fraîche dans l'esprit des joueurs. Au moment de prendre leur décision sur le comportement à adopter, ils prennent donc en compte non seulement l'idée qu'ils se font du comportement général de l'adversaire, mais aussi les cartes qu'il a jouées récemment et qui peuvent donner une indication sur une éventuelle modification de ce comportement.

Pour rendre ce choix possible, nous allons ajouter une information à la mémoire disponible pour les stratégies, à savoir un indicateur mis à jour lors de chaque coup. Cet indicateur ne représente en fait qu'un moyen de compter les cartes `D` (ou `C` selon le point de vue) que l'adversaire a utilisées depuis le début de la partie. La mise en place pratique de cet indicateur est relativement aisée et n'a de sens que pour les stratégies ayant la possibilité de se souvenir de la dernière carte jouée par l'adversaire, c'est-à-dire pour les familles de stratégies pour lesquelles $O \geq 1$.

Au final, pour définir une stratégie de cette famille, il faut donc spécifier :

- la valeur de M,
- la valeur de O,
- l'amorce de la partie,
- la définition du comportement en fonction des historiques possibles pour le cas où l'adversaire a plus souvent trahi que coopéré dans le passé,
- la définition du comportement en fonction des historiques possibles pour le cas où l'adversaire a plus souvent coopéré que trahi dans le passé.

Comme pour les familles de stratégies précédentes, le nom d'une stratégie de cette famille sert de support à la définition de son comportement.

Le nom d'une stratégie est créé par concaténation :

1. du préfixe `bin` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. des cartes à jouer pour chacun des coups de l'amorce dans l'ordre croissant des coups, *i.e* du coup 1 au coup max(M,O) ;
8. de la liste ordonnée des cartes à jouer pour chacune des configurations de la mémoire possible si l'adversaire a plus souvent trahi que coopéré dans le passé, au sens strict.
9. de la liste ordonnée des cartes à jouer pour chacune des configurations de la mémoire possible si l'adversaire a plus souvent coopéré que trahi dans le passé, au sens large.

   Le schéma est ici encore le même que précédemment, aussi bien pour le point 8 que le 9 et il n'est pas répété.

Une seule et même chaîne de caractères est utilisée pour les points 7, 8, et 9. Cette chaîne est donc composée de $(\max(M,O) + 2 \times 2^{M+O})$ soit $(\max(M,O) + 2^{(M+O+1)})$ caractères.

`bin_1_1_cddddcccc` est un exemple de stratégie de la famille `binary`. Elle appartient à la classe des stratégies à mémoire binaire qui n'utilisent que la dernière carte du joueur et la dernière carte de l'adversaire.

Elle peut être interprétée de la façon suivante :

`bin_1_1_cddddcccc`

1. *Au premier coup je joue la carte* C,
2. *ensuite :*
   (a) *si l'adversaire a strictement plus souvent trahi que coopéré depuis le début de la partie et :*
   - *si j'ai joué* C*, et que mon adversaire a joué* C *alors je joue* D
   - *si j'ai joué* C*, et que mon adversaire a joué* D *alors je joue* D
   - *si j'ai joué* D*, et que mon adversaire a joué* C *alors je joue* D
   - *si j'ai joué* D*, et que mon adversaire a joué* D *alors je joue* D

   (b) *sinon :*
   - *si j'ai joué* C*, et que mon adversaire a joué* C *alors je joue* C
   - *si j'ai joué* C*, et que mon adversaire a joué* D *alors je joue* C
   - *si j'ai joué* D*, et que mon adversaire a joué* C *alors je joue* C
   - *si j'ai joué* D*, et que mon adversaire a joué* D *alors je joue* C

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :

C || D D D D || C C C C

On peut remarquer que cette modification n'a pas diminué le pouvoir d'expression de la description des stratégies de la classe `memory`. La stratégie précédente est un autre nom pour `soft_majo`, qui ne pouvait pas être codée dans les familles précédentes. `tit_for_tat` est en fait `bin_1_1_ccdcdcdcd`, `spiteful` est `bin_0_1_cddcd`, `tf2t` est `bin_0_2_cccccdcccd`, etc.

Chaque classe de stratégies de la famille `binary` permet donc de définir $\mathcal{N}_{\texttt{bin}}(\mathrm{M},\mathrm{O})$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi dans l'ensemble {c,d} :

$$\mathcal{N}_{\texttt{bin}}(\mathrm{M},\mathrm{O}) = 2^{\left(\max(\mathrm{M},\mathrm{O})+2^{(\mathrm{M}+\mathrm{O}+1)}\right)} \tag{5.4}$$

La famille `binary_with_dynamic_start`

De même que pour la famille `memory`, les stratégies de la famille `binary` jouent sans réagir au comportement de l'adversaire lors des coups de l'amorce d'une partie. On peut donc lui faire les mêmes reproches. La classe `binary_with_dynamic_start` propose exactement la même solution que celle utilisée pour la classe `memory_with_dynamic_start`.

Le nom d'une stratégie est créé par concaténation :

1. du préfixe `bind` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. de la carte à jouer lors du premier coup ;
8. des cartes à jouer pour chacun des coups suivants de l'amorce dans l'ordre croissant des coups, *i.e* du coup 2 au coup max(M,O) et en fonction de la carte jouée par l'adversaire au coup précedent.
   Le schéma utilisé dans ce cas est exactement le même que celui utilisé dans le cas correspondant pour les stratégies de la famille `memory_with_dynamic_start`, page 84.

9. de la liste ordonnée des cartes à jouer pour chacune des configurations de la mémoire possibles si l'adversaire a plus souvent trahi que coopéré dans le passé, au sens strict.
10. de la liste ordonnée des cartes à jouer pour chacune des configurations de la mémoire possible si l'adversaire a plus souvent coopéré que trahi dans le passé, au sens large.

Le schéma est ici encore le même que précédemment, aussi bien pour le point 9 que le 10.

Une seule et même chaîne de caractères est utilisée pour les points 7, 8, 9 et 10. Cette chaîne est donc composée de $(1 + 2(\max(\mathrm{M},\mathrm{O}) - 1) + 2^{(\mathrm{M}+\mathrm{O}+1)})$ soit $(2\max(\mathrm{M},\mathrm{O}) - 1 + 2^{(\mathrm{M}+\mathrm{O}+1)})$ caractères.

Pour en finir avec la classe des stratégies à mémoire binaire, on peut interpréter la stratégie `bind_0_2_ccdcdcdcdcd` :

`bind_0_2_ccdcdcdcdcd`

1. *Au premier coup je joue la carte* [C],
2. *au second coup si l'adversaire a joué* `C` *je joue* [C], *s'il a joué* `D` *alors je joue* [D]
3. *ensuite :*
   (a) *si l'adversaire a strictement plus souvent trahi que coopéré depuis le début de la partie et :*
      - *si mon adversaire a joué* `C`, *suivi de* `C`, *alors je joue* [C]
      - *si mon adversaire a joué* `C`, *suivi de* `D`, *alors je joue* [D]
      - *si mon adversaire a joué* `D`, *suivi de* `C`, *alors je joue* [C]
      - *si mon adversaire a joué* `D`, *suivi de* `D`, *alors je joue* [D]

   (b) *sinon :*
      - *si mon adversaire a joué* `C`, *suivi de* `C`, *alors je joue* [C]
      - *si mon adversaire a joué* `C`, *suivi de* `D`, *alors je joue* [D]
      - *si mon adversaire a joué* `D`, *suivi de* `C`, *alors je joue* [C]
      - *si mon adversaire a joué* `D`, *suivi de* `D`, *alors je joue* [D]

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :

[C] | [C] [D] || [C] [D] [C] [D] || [C] [D] [C] [D]

On reconnaît une fois de plus dans cette stratégie la stratégie `tit_for_tat`.

Chaque famille de stratégies de la classe `binary_with_dynamic_start` permet donc de définir $\mathcal{N}_{\texttt{bind}}(\mathrm{M},\mathrm{O})$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi dans l'ensemble {`c`,`d`} :

$$\mathcal{N}_{\texttt{bind}}(\mathrm{M},\mathrm{O}) = 2^{\left(2\max(\mathrm{M},\mathrm{O})-1+2^{(\mathrm{M}+\mathrm{O}+1)}\right)} \tag{5.5}$$

Après avoir présenté les stratégies basées uniquement sur l'utilisation de la mémoire, nous allons maintenant décrire les deux dernières familles basées sur un concept fondamental en informatique : les automates.

5.2.3 Les automates de Moore

La première famille de stratégies automate que nous définissons, est celle se basant sur l'idée des automates de Moore. Un rappel de la définition des automates de Moore a été fait dans le chapitre 1, page 15.

En résumé on peut dire que les automates de Moore sont de simples automates d'états finis avec sortie. La sortie de l'automate dépend uniquement de l'état courant de l'automate, via une fonction

de sortie. Les transitions d'états, quant à elles, dépendent des valeurs d'entrées de l'automate et de son état courant.

Ce type d'automate est parfaitement adapté à la définition de stratégies pour le dilemme itéré du prisonnier. On peut, en effet, considérer que les sorties de l'automate correspondent aux cartes à jouer dans chaque circonstance, alors que les entrées correspondent aux cartes déjà jouées.

Ce type d'automates est souvent utilisé dans ce cadre et plus précisément pour l'utilisation d'algorithmes génétiques, voir par exemple [Mil96].

C'est donc cette approche que nous utiliserons ici, en y apportant quelques aménagements.

Tout d'abord, les entrées de l'automate doivent être à valeur dans un alphabet particulier. Nous considérons que les symboles de cet alphabet dépendent des deux mêmes paramètres que pour les familles à mémoires et à mémoires binaires, à savoir M, permettant de savoir combien de cartes le joueur se souvient avoir joué et O, permettant de savoir de combien de cartes de l'adversaire le joueur se souvient.
En bref, cela veut dire que les entrées de l'automate seront les configurations du passé visible. Nous utilisons pour cela exactement le même codage des configurations historiques que celui décrit page 82. La mémoire des stratégies des familles précédentes devient ici simplement la lettre de l'alphabet qui sert d'entrée à l'automate. Pour cela, il faut supposer que les entrées sont composées d'une chaîne de caractères, représentant les deux files d'attentes, qui correspondent aux configurations possibles de la mémoire d'une stratégie. Après chaque coup, cette chaîne de caractères est mise à jour exactement comme le sont les files d'attentes des stratégies `memory` et c'est cette chaîne de caractère qui est donnée en entrée à l'automate. Cette petite gymnastique permet de définir une fonction de transitions pour l'automate qui dépend directement du nombre de cartes du joueur et du nombre de cartes de l'adversaire que l'on veut utiliser.
Ces deux paramètres permettent de définir des classes dans cette famille au même titre que ce que nous avons fait pour les familles précédentes.

Rappelons que la définition d'une transition correspond simplement à la donnée d'un état vers lequel l'automate doit se déplacer en fonction des entrées et de l'état courant.

L'utilisation du même mécanisme de gestion de la mémoire que pour les familles précédentes oblige aussi la spécification du comportement de l'automate tant qu'une entrée n'est pas disponible. Rappelons que, comme pour les familles précédentes, l'automate ne pourra fonctionner uniquement que lorsqu'une entrée sera disponible. Pour qu'une entrée soit disponible, il faut attendre (max(M,O)) coups. Il faut spécifier le comportement de l'automate durant cette période d'*amorce de la partie*.
Pour appliquer la définition formelle d'un automate de Moore, il nous faudrait considérer, que l'automate n'est utilisé qu'après le coup max(M,O) et que son état initial est spécifié. Ceci n'est cependant pas très contraignant, si l'on adapte quel que peu la définition formelle, en considérant simplement que l'automate est initialisé dans un certain état, puis que des changements d'états sont faits pendant la durée de l'amorce de la partie. Ces changements d'états peuvent même être considérés comme des transitions particulières. Nous les appellerons des *transitions d'amorces*.

Ensuite à chaque état doit être attachée une carte. Cette carte qui correspond à la sortie de l'automate quand il est sur un état, devient aussi la carte que le joueur utilisant la stratégie devra jouer. Plus généralement on peut dire que l'état code le comportement courant du joueur, ou plus exactement que le comportement du joueur utilisant la stratégie est fixé par la fonction de sortie.

Un dernier paramètre qui devra être pris en compte d'une manière ou d'une autre sera le nombre d'états de l'automate.

Finalement pour définir complètement une stratégie par un automate de Moore, il nous faut donc spécifier :

1. la valeur de M,
2. la valeur de O,
3. le nombre d'états de l'automate,
4. la fonction de sortie de l'automate,
5. les déplacements d'états pour l'amorçage de la partie,

6. la fonction des transitions d'états.

Comme pour les cas précédents le nom d'une stratégie sert de support à la définition de ce comportement.

Par convention, on représente les cartes du dilemme du prisonnier par des caractères :

– `c` correspond à la coopération (`C`) ;

– `d` correspond à la trahison (`D`).

Les états sont eux aussi représentés par des caractères. Dans le but de simplifier la lecture du nom des stratégies, nous utilisons pour ce codage un décalage du code ASCII. Chaque état d'un automate à n états est étiqueté par un numéro n_i, avec i entre 0 et $n-1$. Le nom de chaque état est représenté par un seul caractère. Ce caractère est le caractère dont le code ASCII est $48 + n_i$. Pour l'état 0 le caractère est alors `0` (code ASCII 48), pour l'état 1 il s'agit du caractère `1` (code ASCII 49), ..., pour l'état 9 le caractère est `9` (code ASCII 57), pour l'état 10 le caractère est `:` (code 58) etc. Le tableau 5.1 fournit les 20 premières correspondances.

Numéro de l'état	0	1	2	3	4	5	6	7	8	9
Code de l'état	`0`	`1`	`2`	`3`	`4`	`5`	`6`	`7`	`8`	`9`
Numéro de l'état	10	11	12	13	14	15	16	17	18	19
Code de l'état	`:`	`;`	`<`	`=`	`>`	`?`	`@`	`A`	`B`	`C`

TAB. 5.1 – *Correspondance du codage des états et du code ASCII*

Cette convention est utilisée dans l'esprit du simulateur, de façon à pouvoir représenter le *génotype* d'une stratégie automate sous la forme d'une chaîne de caractères aisément compréhensible pour des automates de moins de 10 états. Comme nous le verrons plus loin cette contrainte est peu gênante du fait de la taille des classes de stratégies automates, dont la plupart reste encore hors de portée de simulations pour des valeurs de n supérieures à 10.

Le nom d'une stratégie est donc créé par concaténation :

1. du préfixe `moore` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. de la fonction de sortie, représentée simplement par une chaîne de caractères. Cette chaîne doit donc posséder autant de caractères que l'automate a d'états, soit n. Les caractères sont ordonnés par ordre croissant des états. Ils correspondent simplement à la carte qui est la sortie d'un état.
8. du caractère `_` ;
9. des transitions d'amorce à utiliser avant chaque coup lors de l'amorce de la partie. Chaque transition est représentée par un caractère correspondant à l'état vers lequel l'automate doit se déplacer, en considérant que l'état de départ est l'état courant. La chaîne représentant ces transitions comporte donc max(M,O) caractères représentant autant de transitions amorçant l'automate.

   Ces transitions ne sont pas des transitions d'états classiques dans le sens où elles n'utilisent pas d'entrées. Elles correspondent plus à un *changement d'états*, un *déplacement*, qu'à une transition au sens formel du terme.

   Ces transitions peuvent être considérées comme la donnée de l'état initial de l'automate puis des max(M,O) − 1 transitions d'amorces.
10. de la fonction des transitions de l'automate. Les transitions sont ici dans l'ordre croissant des états de départ, puis pour chaque état de départ dans l'ordre lexicographique des représentations

des entrées de l'automate, comme nous l'avons défini page 82 pour les stratégies de la famille `memory`.

En utilisant cet ordre une transition peut être représentée simplement par l'état d'arrivée : l'état de départ et l'entrée sont déductibles de la position dans la chaîne de caractères.

Cette convention permet donc de représenter la fonction de transition de l'automate par une chaîne de $n \times 2^{(\mathrm{M}+\mathrm{O})}$ caractères.

Une seule et même chaîne de caractères est utilisée pour les points 9 et 10. Cette chaîne est donc composée de $(\max(\mathrm{M},\mathrm{O}) + n \times 2^{(\mathrm{M}+\mathrm{O})})$ caractères.

La stratégie `moore_0_2_cd_1001010101`, par exemple est une stratégie de la famille `moore`. Elle appartient à la classe des stratégies utilisant en entrée les valeurs des deux dernières cartes jouées par l'adversaire.

L'automate qui détermine son comportement est constitué de deux états. L'alphabet de sortie de l'automate, comme pour toutes les stratégies de cette classe, est $\Delta = \{\mathtt{C},\mathtt{D}\}$. Le premier de ses états a pour valeur de sortie la coopération et le second la trahison. L'alphabet des symboles d'entrées Σ est alors :

$$\Sigma = \{\mathtt{cc},\mathtt{cd},\mathtt{dc},\mathtt{dd}\}$$

Par exemple, l'entrée `dd` est présentée à l'automate quand l'adversaire a trahi deux fois consécutives dans les deux derniers coups.

L'automate passe dans l'état 1 à chaque fois que l'entrée est incluse dans le sous-ensemble $\{\mathtt{cd},\mathtt{dd}\}$ et dans l'état 0 sinon.

Plus formellement, l'ensemble des états est

$$\mathcal{Q} = \{0,1\}$$

l'alphabet des symboles d'entrée est

$$\Sigma = \{\mathtt{cc},\mathtt{cd},\mathtt{dc},\mathtt{dd}\}$$

l'alphabet des symboles de sortie est

$$\Delta = \{\mathtt{C},\mathtt{D}\}$$

la fonction de sortie de l'automate est

$$\begin{aligned} \lambda_{\mathrm{mo}} : \quad 0 &\mapsto \mathtt{C} \\ 1 &\mapsto \mathtt{D} \end{aligned}$$

la fonction des transitions est

$$\begin{aligned} \delta : \quad (0,\mathtt{cc}) &\mapsto 0 \\ (0,\mathtt{cd}) &\mapsto 1 \\ (0,\mathtt{dc}) &\mapsto 0 \\ (0,\mathtt{dd}) &\mapsto 1 \\ (1,\mathtt{cc}) &\mapsto 0 \\ (1,\mathtt{cd}) &\mapsto 1 \\ (1,\mathtt{dc}) &\mapsto 0 \\ (1,\mathtt{dd}) &\mapsto 1 \end{aligned}$$

La stratégie `moore_0_2_cd_1001010101` peut donc être interprétée comme suit :

`moore_0_2_cd_1001010101`

J'utilise un automate à deux états pour déterminer ce que je dois jouer :

- *si l'automate est dans l'état 0 alors je joue* `C`

– *si l'automate est dans l'état 1 alors je joue* D

1. *Au premier coup je considère que l'automate est dans l'état* 1,
2. *au second coup je considère qu'il est dans l'état* 0,
3. *ensuite si l'automate est dans l'état 0 et*
   – *si mon adversaire a joué* C *puis* C *alors l'automate passe dans l'état* 0
   – *si mon adversaire a joué* C *puis* D *alors l'automate passe dans l'état* 1
   – *si mon adversaire a joué* D *puis* C *alors l'automate passe dans l'état* 0
   – *si mon adversaire a joué* D *puis* D *alors l'automate passe dans l'état* 1

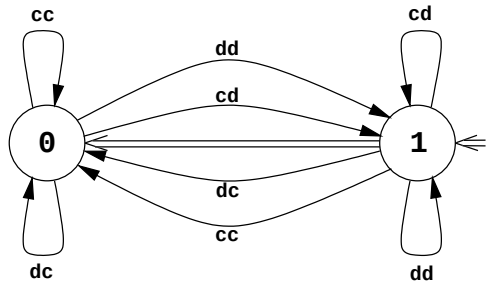
   Si l'automate est dans l'état 1 et
   – *si mon adversaire a joué* C *puis* C *alors l'automate passe dans l'état* 0
   – *si mon adversaire a joué* C *puis* D *alors l'automate passe dans l'état* 1
   – *si mon adversaire a joué* D *puis* C *alors l'automate passe dans l'état* 0
   – *si mon adversaire a joué* D *puis* D *alors l'automate passe dans l'état* 1

À chaque coup je joue ce que me dicte l'état de l'automate.

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :

C D || 1 | 0 || 0 1 0 1 | 0 1 0 1

L'automate correspondant est donc :



Comme nous le verrons dans la section 5.2.5 cette famille inclut les automates de la famille `memory` et donc la remarque faite au moment de la présentation des stratégies de la classe `memory` sur l'expressivité de la classe est encore valide ici. On peut cependant noter que certaines stratégies qui n'étaient pas codables avec les familles précédentes le sont. Les stratégies périodiques en sont un bon exemple. `moore_0_1_ccd_0112200` correspond à la stratégie `per_ccd`.

Chaque classe de stratégies de la famille `moore`, permet donc de définir $\mathcal{N}_{\texttt{moore}}(\mathrm{M},\mathrm{O},n)$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi pour certains dans l'ensemble {c,d} et pour d'autres dans l'ensemble des caractères représentant les états :

$$\mathcal{N}_{\texttt{moore}}(\mathrm{M},\mathrm{O},n) = 2^n \times n^{\max(\mathrm{M},\mathrm{O})} \times \left(n^{2^{(\mathrm{M}+\mathrm{O})}}\right)^n \tag{5.6}$$

5.2.4 Les automates de Mealy

La seconde famille de stratégies automate que nous définissons, est celle se basant sur l'idée des automates de Mealy. Un rappel de la définition des automates de Mealy a été fait dans le chapitre 1, page 15.

En résumé, on peut dire que les automates de Mealy sont de simples automates d'états finis avec sorties. La sortie de l'automate dépend de l'état courant de l'automate et des entrées qui lui sont présentées, via une fonction de sortie. Les transitions d'états, quant à elles, dépendent des valeurs d'entrées de l'automate et de son état courant.

Ce type d'automates est lui aussi bien adapté à la définition de stratégies pour le dilemme itéré du prisonnier pour les mêmes raisons que les automates de MOORE. Ils ne nécessitent guère plus d'aménagements que les automates de MOORE, même si ces aménagements sont légèrement différents.

Les automates de MEALY, comme les automates de MOORE sont eux aussi souvent utilisés pour la représentation des stratégies pour le dilemme du prisonnier dans des approches d'algorithmes génétiques ou évolutionnaires, [Fog93], [Fog95, chapitre 5, section 3, page 205].

La seule différence entre un automate de MOORE et un automate de MEALY réside dans la définition d'une transition. Nous utilisons le même codage en adaptant la représentation pour prendre en compte la définition nouvelle de la fonction de sortie.

En fait, désormais la sortie de l'automate est intimement liée à la fonction de transitions. On peut dire que les deux fonctions n'en font plus qu'une, une transition pouvant être désormais comprise comme :

> « *étant donné l'état courant et une entrée, la sortie est faite par une carte particulière,* ***puis*** *l'état courant de l'automate est modifié.* »

Nous utilisons donc les transitions dans ce sens plus simple à exprimer. Pour représenter une telle transition il nous faudra donc deux caractères, un pour désigner la sortie de l'automate et un autre pour désigner le nouvel état que l'automate doit atteindre.

Pour ce qui est de l'amorce de la partie, nous utilisons exactement le même principe que celui que nous avons utilisé pour les automates de MOORE, à savoir des transitions d'amorçage.

Le nom d'une stratégie est donc créé par concaténation :

1. du préfixe `mealy` ;
2. du caractère `_` ;
3. de la valeur de M ;
4. du caractère `_` ;
5. de la valeur de O ;
6. du caractère `_` ;
7. des transitions d'amorce à utiliser pour chaque coup de l'amorce de la partie. Chaque transition est représentée par un caractère pour la carte à jouer, suivi d'un autre caractère correspondant à l'état vers lequel l'automate doit se déplacer, en considérant que l'état de départ est l'état courant. La chaîne représentant ces transitions comporte donc $2\max(\mathrm{M},\mathrm{O})$ caractères représentant autant de transitions amorçant l'automate.

   Ces transitions ne sont pas des transitions d'états classiques dans le sens où elles n'utilisent pas d'entrées. Elles correspondent plus à un *changement d'états*, un *déplacement*, qu'à une transition au sens formel du terme.
8. de la fonction des transitions de l'automate. Les transitions sont ici dans l'ordre croissant des états de départ, puis pour chaque état de départ dans l'ordre lexicographique des représentations des entrées de l'automate, comme nous l'avons défini page 82 pour les stratégies de la famille `memory`.

   En utilisant cet ordre, une transition peut être représentée simplement par le caractère correspondant à la sortie de l'automate et à l'état d'arrivée : l'état de départ et l'entrée sont déductibles de la position dans la chaîne de caractères.

   Cette convention permet donc de représenter la fonction de transition de l'automate par une chaîne de $(ss2n \times 2^{(\mathrm{M}+\mathrm{O})})$ caractères.

Une seule et même chaîne de caractères est utilisée pour les points 7 et 8. Cette chaîne est donc composée de $(2\max(\mathrm{M},\mathrm{O}) + 2n \times 2^{(\mathrm{M}+\mathrm{O})})$ caractères.

Le nombre d'états n'est pas spécifié dans le nom de la stratégie mais il peut être déduit simplement en utilisant M, O, et la longueur de la chaîne de caractères finale.

La stratégie `mealy_1_1_c0c1d1c1d1c0d0d0d0`, par exemple, est une stratégie de la famille `mealy`. Elle appartient à la classe des stratégies utilisant en entrée la valeur de la dernière carte jouée par le joueur et la dernière carte jouée par l'adversaire.

L'automate qui détermine son comportement est constitué de deux états. L'alphabet de sortie, comme pour les automates de la famille `moore` et tous ceux de la famille `mealy`, est $\Delta = \{\texttt{C},\texttt{D}\}$. L'alphabet des symboles d'entrées Σ est alors :

$$\Sigma = \{\texttt{cc},\texttt{cd},\texttt{dc},\texttt{dd}\}$$

Par exemple, l'entrée `cd` est présentée à l'automate quand le joueur vient de jouer `C`, c'est-à-dire qu'il a coopéré et que l'adversaire a trahi dans le dernier coup. L'automate change d'état à chaque coup.

Plus formellement, l'ensemble des états est

$$\mathcal{Q} = \{0,1\}$$

l'alphabet des symboles d'entrée est

$$\Sigma = \{\texttt{cc},\texttt{cd},\texttt{dc},\texttt{dd}\}$$

l'alphabet des symboles de sortie est

$$\Delta = \{\texttt{C},\texttt{D}\}$$

la fonction de sortie de l'automate est

$$\begin{aligned}
\lambda_{\text{me}} : \quad (0,\texttt{cc}) &\mapsto \texttt{C} \\
(0,\texttt{cd}) &\mapsto \texttt{D} \\
(0,\texttt{dc}) &\mapsto \texttt{C} \\
(0,\texttt{dd}) &\mapsto \texttt{D} \\
(1,\texttt{cc}) &\mapsto \texttt{C} \\
(1,\texttt{cd}) &\mapsto \texttt{D} \\
(1,\texttt{dc}) &\mapsto \texttt{D} \\
(1,\texttt{dd}) &\mapsto \texttt{D}
\end{aligned}$$

la fonction des transitions est

$$\begin{aligned}
\delta : \quad (0,\texttt{cc}) &\mapsto 1 \\
(0,\texttt{cd}) &\mapsto 1 \\
(0,\texttt{dc}) &\mapsto 1 \\
(0,\texttt{dd}) &\mapsto 1 \\
(1,\texttt{cc}) &\mapsto 0 \\
(1,\texttt{cd}) &\mapsto 0 \\
(1,\texttt{dc}) &\mapsto 0 \\
(1,\texttt{dd}) &\mapsto 0
\end{aligned}$$

`mealy_1_1_c0c1d1c1d1c0d0d0d0`

J'utilise un automate à deux états pour déterminer ce que je dois jouer.

1. *Au premier coup je joue* `C` *puis l'automate passe dans l'état* `1`,
2. *ensuite si l'automate est dans l'état 0 et*
   - *si j'ai joué* `C` *et qu'il a joué* `C` *alors je joue* `C` *et l'automate passe dans l'état* `1`
   - *si j'ai joué* `C` *et qu'il a joué* `D` *alors je joue* `D` *et l'automate passe dans l'état* `1`
   - *si j'ai joué* `D` *et qu'il a joué* `C` *alors je joue* `C` *et l'automate passe dans l'état* `1`
   - *si j'ai joué* `D` *et qu'il a joué* `D` *alors je joue* `D` *et l'automate passe dans l'état* `1`

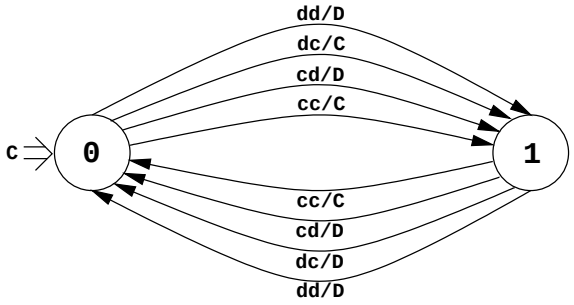
Si l'automate est dans l'état 1 et

- *si j'ai joué* C *et qu'il a joué* C *alors je joue* [C] *et l'automate passe dans l'état* [0]
- *si j'ai joué* C *et qu'il a joué* D *alors je joue* [D] *et l'automate passe dans l'état* [0]
- *si j'ai joué* D *et qu'il a joué* C *alors je joue* [D] *et l'automate passe dans l'état* [0]
- *si j'ai joué* D *et qu'il a joué* D *alors je joue* [D] *et l'automate passe dans l'état* [0]

D'où finalement on obtient la partie intéressante du génotype en concaténant le tout :

[C][0] || [C][1][D][1][C][1][D][1] | [C][0][D][0][D][0][D][0]

L'automate correspondant est donc :



Chaque classe de stratégies de la famille `mealy`, permet donc de définir $\mathcal{N}_{\texttt{mealy}}(\mathrm{M},\mathrm{O},n)$ stratégies complètement différentes, puisque chaque caractère de la chaîne du génotype d'une stratégie peut être choisi pour certains dans l'ensemble {c,d} et pour d'autres dans l'ensemble des caractères représentant les états :

$$\mathcal{N}_{\texttt{mealy}}(\mathrm{M},\mathrm{O},n) = (2n)^{\max(\mathrm{M},\mathrm{O})} \times \left((2n)^{2^{(\mathrm{M}+\mathrm{O})}}\right)^n \tag{5.7}$$

Nous avons maintenant défini toutes les familles de stratégies utilisées pour faire des simulations sur des classes complètes. Avant d'examiner les expériences qui ont été faites, nous allons décrire les relations qui existent entre les différentes familles ainsi que quelques classes particulières.

5.2.5 Relations entre classes

Il est important de noter que les familles décrites précédemment permettent de définir un nombre important de classes complètes différentes. La taille de ces classes est un facteur important puisqu'elle est à la base de leur construction.

La diversité des comportements est assurée par la structure des familles décrites et surtout sur les idées fondatrices de celles-ci. La principale idée que nous avons utilisée consiste en une limitation de la vision du passé. De nombreux travaux sont basés sur la même idée naturelle.

Un des avantages de ces structures est que le choix des stratégies n'est a priori pas subjectif. Moins en tout cas que dans le cas d'une simulation impliquant des stratégies décrites par des joueurs humains, comme par exemple les expériences d'Axelrod, cf. [Axe92], ou de Delahaye et Mathieu, cf. [DM93]. Cela ne signifie pas que ces expériences ne soient pas intéressantes, loin de là, mais plutôt que la validité de leurs résultats peut toujours être mise en doute du fait non seulement du faible nombre de stratégies utilisées, mais aussi des collusions possibles entre participants qui ne sont pas évitables. Dans le cas des classes complètes, ces mêmes arguments sont difficilement utilisables.

Rappelons qu'une classe complète correspond à l'ensemble de toutes les stratégies définissables à partir d'une structure particulière. Dans notre cas les différentes structures sont les familles :

- `memory`,
- `memory_with_dynamic_start`,
- `memory_with_limited_dynamic_start`,
- `binary`,

– `binary_with_dynamic_start`,
– `moore`,
– `mealy`

Afin de mieux identifier les différentes classes complètes de stratégie nous désignons les classes complètes par une construction proche de celle utilisée pour le nommage de leurs stratégies. En bref, les classes complètes seront représentées par la concaténation du préfixe représentant leur famille et des différents paramètres les caractérisant. Pour les classes des familles de stratégies à mémoire, ou à mémoire binaires, ces paramètres sont uniquement les valeurs de M et O ; pour les classes issues des familles de stratégies automates, ces paramètres incluent en plus le nombre d'états des automates. Une classe complète ne pourra donc pas inclure des stratégies ayant un nombre d'états différents.

Voici quelques exemples de classes complètes :

– `mem_0_1`, est la classe complète des stratégies de la famille `memory` avec M=0, et O=1 ;
– `bind_2_1`, est la classe complète des stratégies de la famille `binary_with_dynamic_start` avec M=2 et O=1 ;
– `moore_0_1_2`, est la classe complète des stratégies de la famille `moore` avec M=0, O=1 et $n = 2$, c'est-à-dire à 2 états.

Il est intéressant de noter que **toutes** les stratégies d'une même classe complète ont des comportements différents. Il se peut cependant que ces comportements semblent identiques dans de nombreux cas, face à une stratégie monotone comme `all_c` ou `all_d` par exemple.

La *taille* d'une classe complète est simplement le nombre de stratégies qui composent la classe. Le tableau 5.2 récapitule les tailles des différentes classes complètes basées sur les familles disponibles en fonction de leurs paramètres.

Classe	Taille
`mem`$\_m\_o$	$2^{\left(\max(m,o)+2^{(m+o)}\right)}$
`memd`$\_m\_o$	$2^{\left(2\max(m,o)-1+2^{(m+o)}\right)}$
`memld`$\_m\_o$	$2^{\left(\max(m,o)+1+2^{(m+o)}\right)}$
`bin`$\_m\_o$	$2^{\left(\max(m,o)+2^{(m+o+1)}\right)}$
`bind`$\_m\_o$	$2^{\left(2\max(m,o)-1+2^{(m+o+1)}\right)}$
`moore`$\_m\_o\_n$	$2^n \times n^{\max(m,o)} \times \left(n^{2^{(m+o)}}\right)^n$
`mealy`$\_m\_o\_n$	$(2n)^{\max(m,o)} \times \left((2n)^{2^{(m+o)}}\right)^n$

TAB. 5.2 – *Tailles des classes complètes.*

Les tableaux 5.3, 5.4 et 5.5 indiquent la taille de certaines classes complètes.

Classe	Taille
`mem_0_1`	8
`mem_0_2`	64
`mem_0_3`	2 048
`mem_0_4`	1 048 576
`mem_1_1`	32
`mem_1_2`	1 024
`mem_1_3`	524 288

Classe	Taille
`memd_0_2`	128
`memd_0_3`	8 192
`memd_0_4`	8 388 608
`memd_1_2`	2 048
`memd_1_3`	2 097 152

Classe	Taille
`memld_0_3`	4 096
`memld_0_4`	2 097 152

TAB. 5.3 – *Taille des classes complètes des familles à mémoires seules.*

L'étude des valeurs de ces tables indique clairement les classes dont les tailles sont suffisamment raisonnables pour pouvoir être utilisées dans des simulations d'évolution écologique. Pour chaque variation d'un des paramètres un exemple de classe inaccessible du fait de sa trop grande taille est

Classe	Taille
bin_0_1	32
bin_0_2	1 024
bin_0_3	524 288
bin_1_1	512
bin_1_2	262 144

Classe	Taille
bind_0_2	2 048
bind_0_3	2 097 152
bind_1_2	524 288

TAB. 5.4 – *Taille des classes complètes des familles à mémoires binaires.*

Classe	Taille
moore_0_1_2	128
moore_0_2_2	4 096
moore_0_3_2	2 097 152
moore_1_1_2	2 048
moore_1_2_2	1 048 576
moore_0_1_3	17 496
mealy_0_1_2	1 024
mealy_0_1_3	279 936
mealy_0_2_2	1 048 576
mealy_1_1_2	262 144

TAB. 5.5 – *Taille des classes complètes des familles automates*

indiquée dans les cellules grisées. L'inaccessibilité de ces classes est directement liée aux limitations de notre simulateur logiciel et de la puissance de calcul, mais aussi de stockage, des ordinateurs à notre disposition.

Il est clair que toute les classes ne sont pas équivalentes, mais il existe cependant des relations entre les stratégies de certaines classes et plus largement entre certaines classes.

Deux classes complètes de stratégies de deux familles différentes sont dites **correspondantes** si elles partagent leurs valeurs de paramètres :

- M, et O pour les classes à mémoires et à mémoires binaires,
- M, O, et n (le nombre d'états) pour les stratégies automates.

Proposition 5.1 *Toutes les stratégies des classes à amorce fixe sont incluses dans les classes à amorce dynamique correspondantes.*

La raison en est simple comme le montre le cas de la stratégie `mem_0_2_ccdddd` qui commence par coopérer deux fois puis trahit toujours. Cette stratégie a un comportement parfaitement identique à la stratégie `memd_0_2_cccdddd`. En fait, toutes les stratégies à amorce fixe d'une classe particulière sont incluses dans la classe à amorce dynamique équivalente : elles n'utilisent tout simplement pas la dynamicité de l'amorce.

On a donc :

$$\texttt{mem\_}m\texttt{\_}o \subset \texttt{memd\_}m\texttt{\_}o$$
$$\texttt{bin\_}m\texttt{\_}o \subset \texttt{bind\_}m\texttt{\_}o$$

Proposition 5.2 *Toutes les stratégies des classes à mémoires simples sont incluses dans les classes à mémoire binaires correspondantes.*

Ici encore l'explication est évidente. Les stratégies de la famille à mémoires binaires ajoutent un mécanisme pour la détermination du comportement. Certaines stratégies des classes de cette

famille peuvent ne pas utiliser ce mécanisme. La stratégie `mem_0_1_ccd` a le même comportement que `bin_0_1_ccdcd`. Il s'agit en fait dans les deux cas de la stratégie `tit_for_tat`.

On a :

$$\begin{aligned} \texttt{mem\_}m\texttt{\_}o &\subset \texttt{bin\_}m\texttt{\_}o \\ \texttt{memd\_}m\texttt{\_}o &\subset \texttt{bind\_}m\texttt{\_}o \end{aligned}$$

On peut évidemment combiner les propositions 5.1 et 5.2 et obtenir de ce fait :

$$\texttt{mem\_}m\texttt{\_}o \subset \texttt{bind\_}m\texttt{\_}o$$

Proposition 5.3 *Toutes les stratégies des classes à mémoire simples et binaires sont incluses dans les classes de stratégies automates, au sens de* Mealy*, correspondantes.*

En effet, il est évident que toutes les stratégies des classes à mémoire simple sont incluses dans les classes de stratégies automates de Mealy correspondantes à un seul état, du fait de la définition de la fonction de sortie des automates de Mealy. De plus on comprend aisément également que toutes les stratégies à mémoires binaires d'une classe particulière sont incluses dans une classe à automates de Mealy correspondante qui compte deux états. Un état peut être utilisé pour le cas où l'adversaire a plus souvent trahi que coopéré et l'autre pour le cas contraire. L'automate permet en quelque sorte de coder l'indicateur binaire.

Or il est tout aussi trivial de voir que toutes les stratégies d'une classe d'automates de Mealy à n états sont incluses dans la classe correspondante à $n+1$ états. Plus largement on a même :

Proposition 5.4 *Toutes les stratégies d'une classe d'automates à n états sont incluses dans les classes d'automates correspondantes à m états, avec $n \leq m$.*

Une fois de plus, l'idée essentielle est que chaque famille ajoutant un mécanisme de choix par rapport à une autre permet de décrire des classes complètes incluant toutes les classes de la seconde. Une classe complète est faite de toutes les stratégies d'une classe, donc entre autre des stratégies qui n'utilisent pas ce nouveau mécanisme.

Cette remarque est importante puisqu'elle permet de voir que certaines classes sont des restrictions d'autres et que, plus largement, faire des simulations incluant deux classes complètes *cousines* n'a que peu d'intérêt. En revanche, la modification des paramètres M, et O, assure quasiment à deux classes de la même famille de n'avoir aucune stratégie au même comportement. Il se peut évidemment qu'en moyenne et face à des individus particuliers les comportements semblent identiques, mais au sein des classes les comportements ne peuvent être les mêmes face à toutes les stratégies.

Pour conclure la figure 5.1 représente sous la forme d'un schéma les différentes relations possibles entre les classes correspondantes des quatre familles de stratégies définies dans ce chapitre.

5.3 Expériences sur des classes complètes

Nous allons maintenant pourvoir présenter certains des résultats des différentes simulations effectuées sur des classes complètes de stratégies. Certains résultats présentés ici ont été publiés dans [BDM98].

5.3.1 Description des expériences

La construction des différentes classes complètes de stratégies a toujours été guidée par l'évolution du simulateur logiciel. Ce simulateur permet désormais d'utiliser n'importe laquelle des stratégies d'une des familles définies dans ce chapitre tout simplement en fournissant la liste des stratégies à

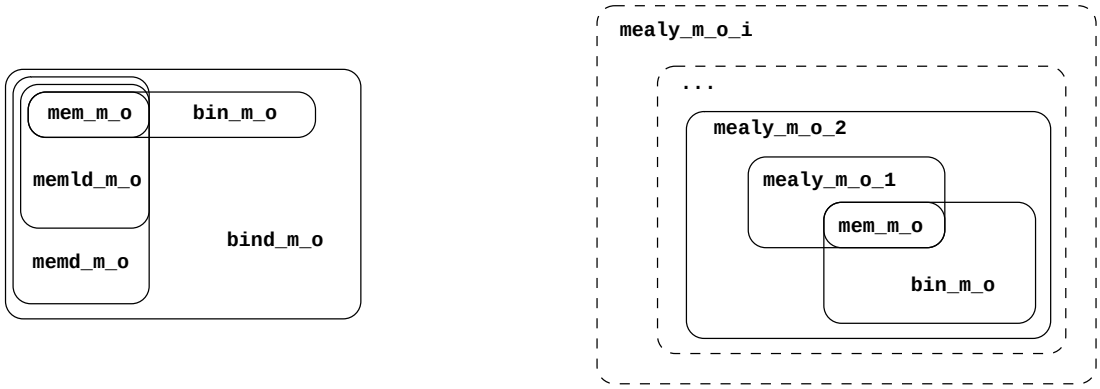


FIG. 5.1 – *Les relations entre classes complètes pour M=m et O=o.*

utiliser via leur nom [29]. Les premières versions du simulateur, dues au travail de Philippe MATHIEU, ne permettaient pas de gérer dynamiquement les classes complètes. La version du simulateur utilisée pour les expériences présentées ici, utilise les mêmes algorithmes mais a subi de nombreuses modifications et réécritures au cours du temps.

Les algorithmes de base utilisés tant pour le calcul de la matrice de gains, c'est-à-dire de la matrice des résultats d'un tournoi, que pour le calcul des évolutions écologiques sont décrits dans l'annexe B, page 133. Les algorithmes utilisés pour les différentes familles de stratégies définies dans ce chapitre sont, quant à eux, donnés en annexe A, page 131. La présence de ces données nous semble indispensable dans cette thèse afin que les expériences et résultats présentés ici puissent, le cas échéant être reproduits et vérifiés.

Par exemple la classe complète `memd_0_3` qui comporte 8 192 stratégies est aujourd'hui encore hors de portée, non du simulateur, mais des ordinateurs à notre disposition. En effet, cette classe comporte 8 192 stratégies. Sachant, par expérience, qu'une évolution écologique sur un si grand nombre de stratégies a des chances de durer plus de 10 000 cycles, il faut donc que le simulateur réserve un espace mémoire capable de stocker $8\,192 \times 10\,000$ entiers correspondant aux différentes étapes de l'évolution. Sachant que sur les machines dont nous disposons les entiers sont stockés sur 4 octets, il nous aurait fallu des machines permettant de réserver plus de 320 Méga octets pour un seul programme. Ces machines n'étant pas encore à notre disposition, et n'ayant pas souhaité utiliser des méthodes à base mémoire de masses à cause de la lenteur des accès, nous nous sommes contentés de simuler des évolutions écologiques n'impliquant pas plus de 4 500 stratégies.

Néanmoins des solutions basées sur le calcul distribué, à base de systèmes multi-agents sont en cours de développement et nous laissent de bons espoirs pour le futur proche.

Ceci étant dit le tableau 5.6 dresse un récapitulatif des différentes simulations d'évolutions écologiques effectuées. Au total 23 432 stratégies ont été créées.

À chaque fois le protocole a été le même, à savoir :

1. Calculer la matrice de gains de la classe complète ;
2. Calculer une évolution sur la classe complète ;
3. Calculer une évolution sur la classe complète plus la stratégie `gradual` ;
4. Calculer une évolution sur la classe complète plus la stratégie `bad_bet`.

Les paramètres du dilemme utilisés sont ceux du dilemme classique, comme définis page 41, à savoir : $T = 5$, $R = 3$, $P = 1$, $S = 0$. Chaque partie a duré 1 000 coups.

Nous présentons maintenant certains des résultats des simulations effectuées.

29. L'utilisation des caractères génériques `*` et `%` facilitant grandement cette tâche. Pour plus de précisions on pourra se reporter à l'aide du logiciel disponible sur le site web du projet PRISON à l'adresse `http://www.lifl.fr/IPD`

mem_0_1	:	8 stratégies
mem_0_2	:	64 stratégies
mem_0_3	:	2 048 stratégies
mem_1_1	:	32 stratégies
mem_1_2	:	1 024 stratégies
mem_2_1	:	1 024 stratégies
memd_0_2	:	128 stratégies
memd_1_2	:	2 048 stratégies
memd_2_1	:	2 048 stratégies
memld_0_3	:	4 096 stratégies
bin_0_1	:	32 stratégies
bin_0_2	:	1 024 stratégies
bin_1_1	:	512 stratégies
bind_0_2	:	2 048 stratégies
moore_0_1_2	:	128 stratégies
moore_0_2_2	:	4 096 stratégies
moore_1_1_2	:	2 048 stratégies
mealy_0_1_2	:	1 024 stratégies

TAB. 5.6 – *Liste des évolutions écologiques de classes complètes effectuées.*

5.3.2 Résultats des expériences

Sur l'ensemble des courbes présentées dans la suite de ce chapitre, seule la courbe d'évolution des 20 meilleures stratégies est représentée.

L'exemple de mem_0_1

Voyons tout d'abord les résultats de la classe complète la plus simple et la plus petite, c'est-à-dire la classe mem_0_1. La figure 5.2 représente l'évolution écologique de cette classe complète.

La première stratégie est la stratégie mem_0_1_ccd, qui correspond en fait à tit_for_tat. La seconde stratégie est la stratégie mem_0_1_dcd, qui correspond en fait à mistrust. La troisième stratégie est la stratégie mem_0_1_cdd.

Les résultats de cette évolution de classe complète ne sont pas très surprenants. En effet parmi les huit stratégies présentes, si l'on fait abstraction de la carte jouée au premier coup, on ne dénombre que quatre comportements différents, à savoir, la trahison continue, la coopération continue, le comportement « *à la* » tit_for_tat et un comportement inverse de tit_for_tat, d'exploitation réciproque. La supériorité de tit_for_tat est alors indiscutable et prévisible. En effet seul tit_for_tat possède les critères de qualité de réactivité, de pardon et de bienveillance dans cette classe.

Certes mem_0_1_dcd en possède deux, mais le fait que cette stratégie prenne l'initiative de trahir et qui plus est au premier coup, l'handicape forcément, ne serait-ce que lorsqu'un joueur utilisant cette stratégie, rencontre un joueur utilisant la même stratégie. En moyenne, on voit bien que $V(\texttt{mem\_0\_1\_dcd}|\texttt{mem\_0\_1\_dcd}) = P$, ici 1, est la cause de la disparition de cette stratégie. Jusqu'à la génération 12, cette stratégie arrive à survivre du fait de la présence de stratégies méchantes dans la population, qui font de faibles scores non seulement face à elles-mêmes, mais aussi face à mem_0_1_dcd. Ensuite, elle se retrouve seule face à tit_for_tat qui joue très bien contre elle même et moyennement bien face à mem_0_1_dcd. Aucune chance n'étant donnée à la coopération tit_for_tat prolifère tranquillement et anéantit sa cousine.

Pour ce qui est de la suite du classement et plus généralement de l'allure de la courbe on se retrouve dans une situation qui se répétera assez souvent dans le reste des expériences, mais avec des facteurs de vitesses différents. Peu de dynamiques complexes apparaissent ici. L'histoire est quasiment toujours la même.

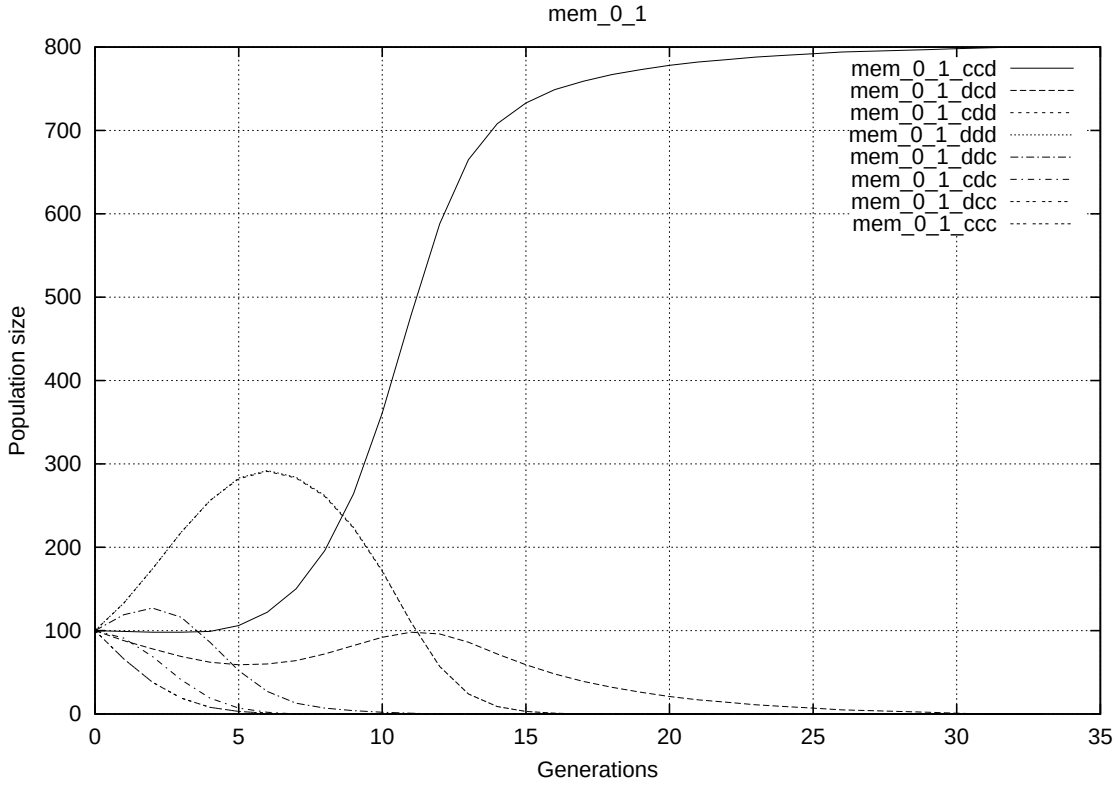


FIG. 5.2 – *Évolution de la classe* `mem_0_1`.

Au début, de nombreuses stratégies profitent de la présence dans la population de stratégies se laissant exploiter avec facilité. Au fur et à mesure de la prolifération des exploiteuses les exploitées ont tendance à disparaître. Pendant ce temps, certaines stratégies au comportement en moyenne, proche de celui de `tit_for_tat` résistent à l'exploitation et maintiennent la taille de leur population à une valeur à peu près fixe.
Dans le cas de `mem_0_1`, les exploitées sont les stratégies ayant comme deux derniers caractères de leur génotype `cc`, les exploiteuses étants les stratégies `mem_0_1_ddd`, `mem_0_1_cdd`, mais aussi les stratégies `mem_0_1_ddc` et `mem_0_1_cdc`.

La baisse des stratégies exploitées entraîne mécaniquement la baisse des exploiteuses. Parmi ces dernières on voit alors ressurgir un certain nombre de comportements pendant une courte période. Ces comportements correspondent en fait à des stratégies ayant les bonnes caractéristiques de réactivité et de pardon mais qui n'ont pas la possibilité de se les appliquer, puisqu'elles sont méchantes, ou en tout cas méfiantes et prennent l'initiative d'une trahison dans l'amorce de la partie. Pendant ce temps leurs stratégies cousines, aux bonnes qualités prolifèrent avec une vigueur extraordinaire du simple fait qu'elles résistent toujours aux exploiteuses, qu'elles s'appliquent à elles-mêmes les bonnes recettes, que le nombre de comportements diminue et que donc elle peuvent se répandre à leur guise. Il faut remarquer que parmi les exploiteuses, on trouve des stratégies qui savent profiter de la faiblesse des autres, mais ne savent pas se prémunir de leur propre faiblesse. C'est ici le cas pour `mem_0_1_ddc` par exemple. Certaines exploitées réussissent à mieux se protéger contre les exploiteuses en perte de vitesse et regagnent un peu de terrain.

Dans une troisième phase les stratégies méfiantes disparaissent complètement, ou en tout cas faiblissent pour la simple raison qu'elles se comportent mal contre elles-mêmes et que les survivantes ne pardonnent cette méfiance qu'à moitié, dans le sens où la coopération n'arrive pas à s'installer.

Cette évolution est assez significative de toutes les autres et les vagues que l'on voit apparaître sur la courbe de la figure 5.2 seront présentes dans presque toutes les autres évolutions. Elles corres-

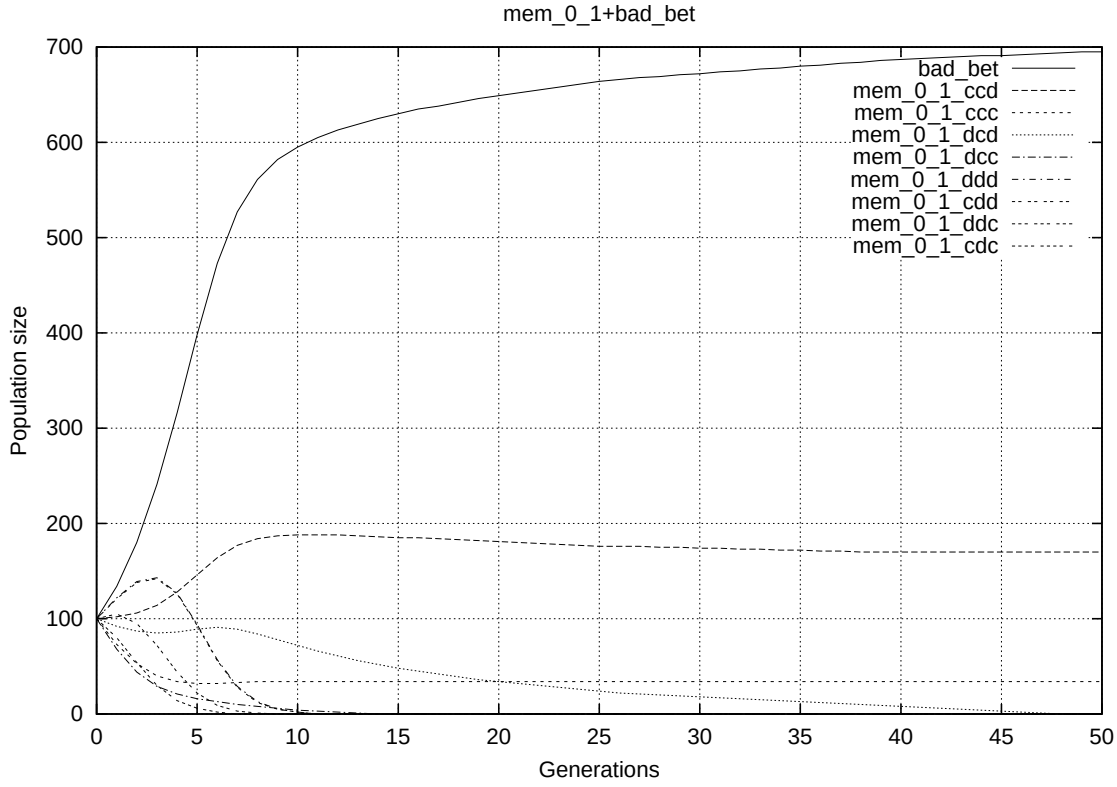


FIG. 5.3 – *Évolution de* `bad_bet` *dans la classe* `mem_0_1`.

pondent en fait à des populations qui profitent momentanément d'autres populations. Puis au fur et à mesure de la disparition de stratégies, d'autres systèmes d'exploitations se mettent en place, mais disparaissent au fur et à mesure, etc.

Sur le long terme la trahison ne paie donc que très rarement quand le comportement de l'adversaire est pris en compte de manière plus significative que le comportement du joueur, c'est-à-dire quand $M < O$. Ces variations en vagues sont sensibles plus ou moins rapides et visibles, mais toujours présentes. La proximité des comportements des stratégies d'une même classe en est sans doute la cause. Les cartes jouées durant l'amorce de la partie servent souvent de discriminant assez tard dans l'évolution, une fois que les comportements clairement non optimaux ont été éliminés.

On imagine assez facilement qu'au début de l'évolution, les animaux les plus faibles se font chasser par tout un tas d'autres animaux plus forts. Après la disparition des plus faibles d'autres niveaux hiérarchiques ont alors pu apparaître. Le terme de l'évolution montre alors la prolifération d'individus suffisamment fort pour se défendre contre les agressions et suffisamment malins pour éviter de détruire trop rapidement leur source de force.

La figure 5.3 représente l'évolution écologique de la classe `mem_0_1` dans laquelle on a rajouté la stratégie `bad_bet`.

Cette courbe est elle aussi très représentative du comportement que `bad_bet` peut avoir face aux différentes classes complètes. Très vite, `bad_bet` s'impose comme une des stratégies à la plus forte progression. Elle sort du lot, laissant les autres stratégies à leurs querelles, comme si elle avait tout de suite compris la manière d'agir avec les comportements présents dans la classe. C'est ici la force d'adaptation de `bad_bet` qui fait la différence. Mieux que les comportements simples « *à la* » `tit_for_tat`, `bad_bet` réussit à adapter son comportement au niveau d'agressivité de son adversaire. Elle n'essaie pas de le forcer à coopérer, mais tente plutôt de punir son adversaire en choisissant le comportement qui lui rapporte le plus, c'est-à-dire en ne prenant plus en compte l'intérêt du groupe mais uniquement le sien. Sa grande force ici vient du fait qu'elle peut facilement identifier le meilleur

comportement à adopter face à son adversaire qui n'a qu'une vue limitée du passé. L'adaptation à l'adversaire est la grande force de `bad_bet`.

`gradual` y réussira aussi souvent mais son degré d'adaptation se fait sur une période plus longue et de manière continue. Elle s'adapte d'une seule manière à son adversaire en trahissant de plus en plus de telle sorte qu'avec les stratégies de ces classes qui n'ont qu'une vue limitée du passé elle se fait rapidement passer pour une méchante et la cause est alors entendue. Au lieu de s'adapter comme le fait `bad_bet`, elle se fond au paysage. L'exemple de la classe `mem_1_2` est significatif en cela. La figure 5.4 représente l'évolution de `gradual` dans la classe complète `mem_1_2`.

La meilleure stratégie de cette évolution est également la meilleure stratégie de la classe complète `mem_1_2` : `mem_1_2_cccdcddcdd`. Cette stratégie peut être interprétée comme :

`mem_1_2_cccdcddcdd`

1. *Au premier coup je joue la carte* `C`,
2. *au second coup je joue la carte* `C`,
3. *ensuite :*
   - *si j'ai joué* `C`*, et que mon adversaire a joué* `C` *puis* `C` *alors je joue* `C`
   - *si j'ai joué* `C`*, et que mon adversaire a joué* `C` *puis* `D` *alors je joue* `D`
   - *si j'ai joué* `C`*, et que mon adversaire a joué* `D` *puis* `C` *alors je joue* `C`
   - *si j'ai joué* `C`*, et que mon adversaire a joué* `D` *puis* `D` *alors je joue* `D`
   - *si j'ai joué* `D`*, et que mon adversaire a joué* `C` *puis* `C` *alors je joue* `D`
   - *si j'ai joué* `D`*, et que mon adversaire a joué* `C` *puis* `D` *alors je joue* `C`
   - *si j'ai joué* `D`*, et que mon adversaire a joué* `D` *puis* `C` *alors je joue* `D`
   - *si j'ai joué* `D`*, et que mon adversaire a joué* `D` *puis* `D` *alors je joue* `D`

Globalement, cette stratégie est bienveillante. Dès qu'elle est trahie, elle punit son adversaire. Ensuite elle tente d'exploiter son adversaire s'il semble accepter sa punition, ou continue à trahir si l'adversaire en fait de même. Elle ne se résout à recoopérer que si l'adversaire lui fait comprendre qu'il n'a pas l'intention de se laisser faire. On comprend bien que cette stratégie adapte son comportement d'une manière beaucoup plus souple et surtout beaucoup plus rapide que `gradual`. De ce fait elle se répand beaucoup plus vite que les autres, un peu comme `bad_bet` le fait dans de nombreuses classes.

Cas singuliers

Toutes les évolutions de classes complètes se ressemblent. Seules quelques unes sortent du lot et exhibent des comportements étonnants. Tout d'abord, on peut noter que dans certaines classes complètes beaucoup de stratégies ne disparaissent pas complètement et qu'un équilibre est atteint assez vite. C'est le cas par exemple de la classe `bin_1_1` (voir figure 5.5).

Cette particularité, rencontrée souvent dans les classes des familles de stratégies à mémoires binaires peut s'expliquer justement par l'aspect binaire de la mémoire. En effet plus de la moitié des stratégies de ces classes sont bienveillantes, à savoir qu'elles ne prennent jamais l'initiative de la trahison.

Cette classe est également intéressante car c'est la seule dans laquelle la stratégie `gradual` se comporte mieux que la stratégie `bad_bet`. L'explication en est cependant assez simple. Les stratégies de la famille `binary` utilisent dans leur comportement d'abord le celui de l'adversaire depuis le début de la partie. C'est-à-dire qu'elles privilégient le long terme au cours terme. Or, comme nous l'avons déjà dit la force de `bad_bet` dans nos classes complètes est de réagir et de s'adapter à son adversaire vite et souvent, contrairement à `gradual`, qui ne s'adapte que d'une seule façon et sur le long terme. Dans cette famille particulière, `gradual` prend alors un avantage sur `bad_bet`.

On peut noter que pour les évolutions de la classe `mem_1_2` et la classe `memd_1_2` les cinq premières stratégies sont identiques. La figure 5.6 illustre ces deux évolutions.

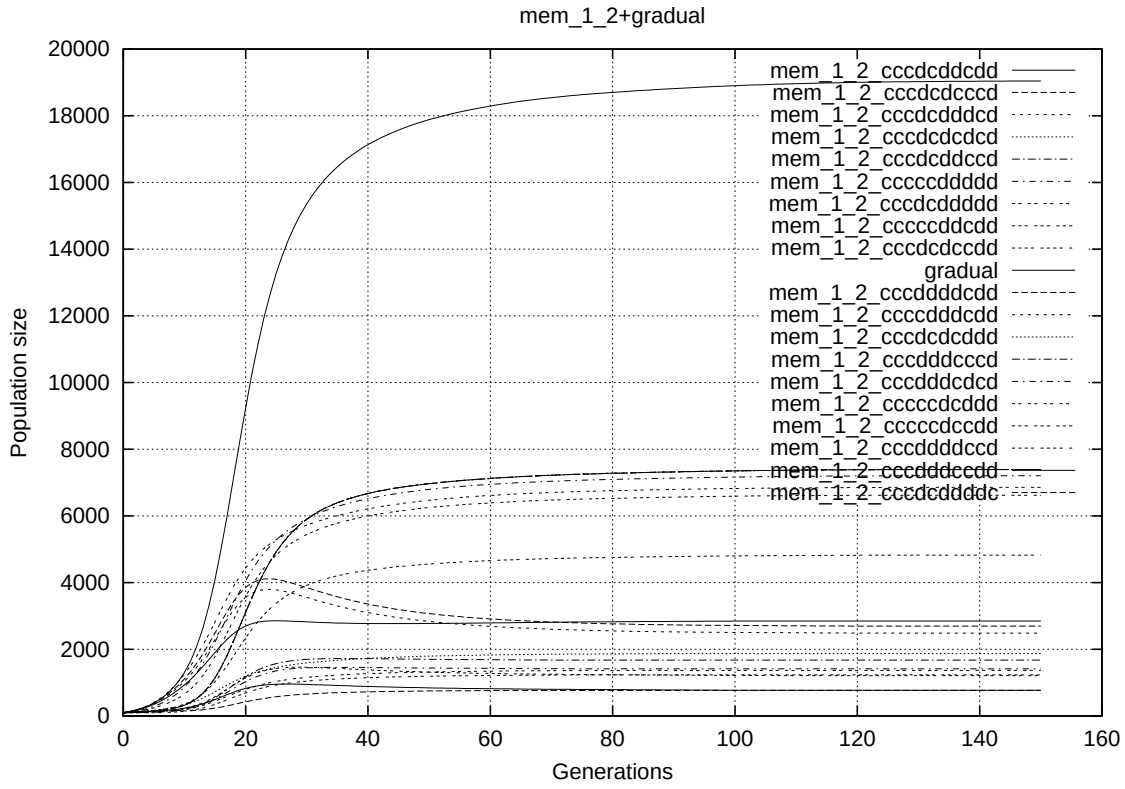


FIG. 5.4 – *Évolution de* `gradual` *dans la classe* `mem_1_2`.

On peut donc raisonnablement penser que la dynamicité apporte peu dans ce cas et que finalement les coups de l'amorce n'ont pas une importance capitale dans les comportements des stratégies de cette classe.

Enfin, un dernier cas intéressant se présente pour les classes dans lesquelles M > O. Les classes `mem_2_1` et `memd_2_1` correspondent à ce cas. L'évolution de la classe `mem_2_1` est représentée sur la figure 5.7.

La meilleure stratégie est exactement la même dans les deux cas. Un soupçon de plus en défaveur de la dynamicité dans les coups de l'amorce. Cette stratégie est `mem_2_1_dccdcdcddd`, qui peut être interprétée comme :

`mem_2_1_dccdcdcddd`

1. *Au premier coup je joue la carte* `D`,
2. *au second coup je joue la carte* `C`,
3. *ensuite :*
   - *si j'ai joué* `C`, *puis* `C` *et que mon adversaire a joué* `C` *alors je joue* `C`
   - *si j'ai joué* `C`, *puis* `C` *et que mon adversaire a joué* `D` *alors je joue* `D`
   - *si j'ai joué* `C`, *puis* `D` *et que mon adversaire a joué* `C` *alors je joue* `C`
   - *si j'ai joué* `C`, *puis* `D` *et que mon adversaire a joué* `D` *alors je joue* `D`
   - *si j'ai joué* `D`, *puis* `C` *et que mon adversaire a joué* `C` *alors je joue* `C`
   - *si j'ai joué* `D`, *puis* `C` *et que mon adversaire a joué* `D` *alors je joue* `D`
   - *si j'ai joué* `D`, *puis* `D` *et que mon adversaire a joué* `C` *alors je joue* `D`
   - *si j'ai joué* `D`, *puis* `D` *et que mon adversaire a joué* `D` *alors je joue* `D`

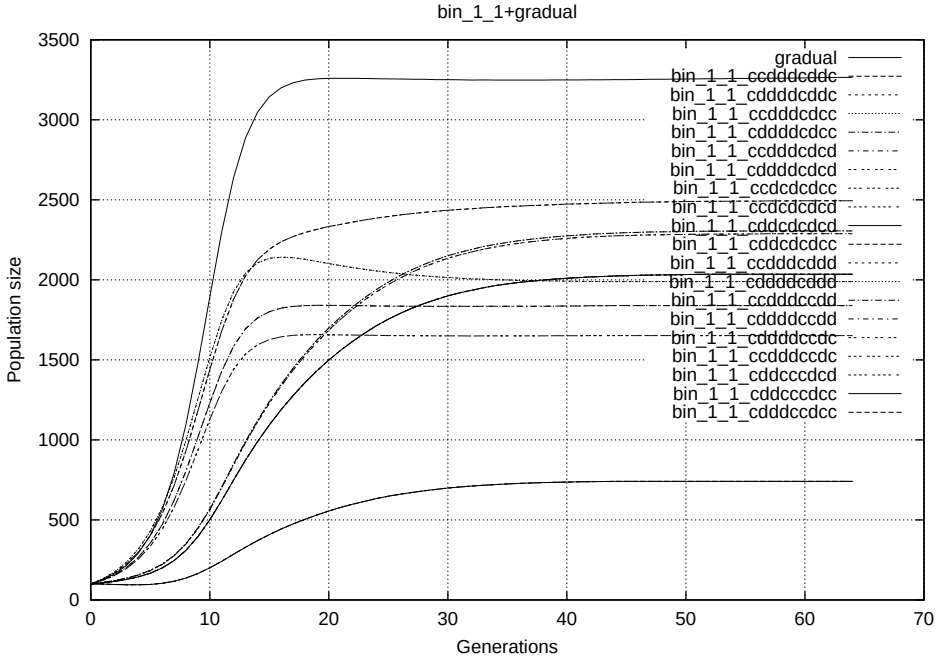


FIG. 5.5 – *Évolution de* `gradual` *dans la classe complète* `bin_1_1`.

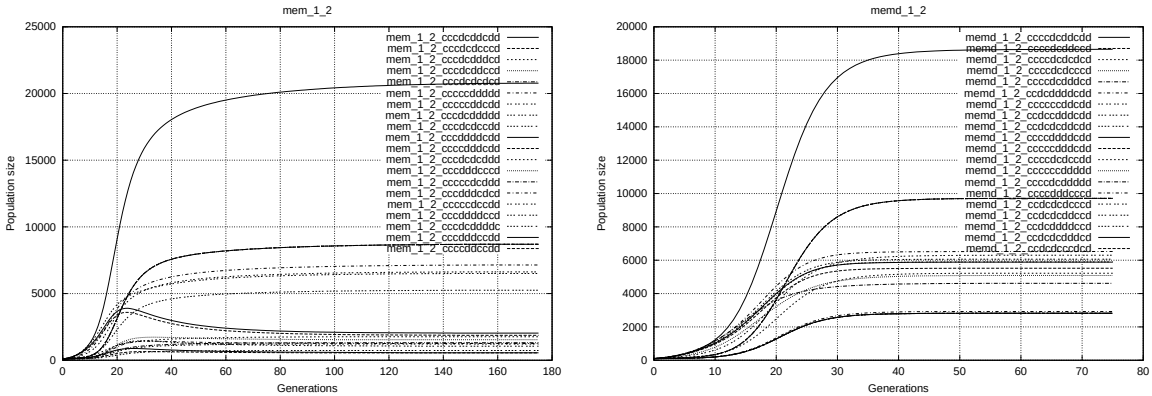


FIG. 5.6 – *Évolutions comparées des classes* `mem_1_2` *et* `memd_1_2`.

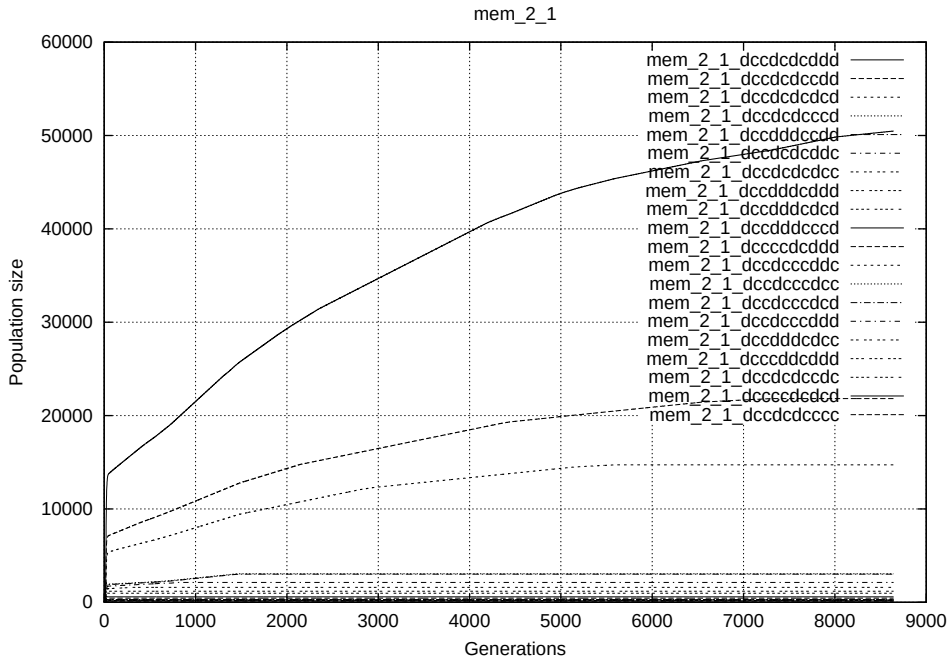


FIG. 5.7 – *Évolutions de la classe complète* `mem_2_1`

Si l'on fait abstraction des coups d'amorce, le comportement de cette stratégie est de toujours jouer exactement comme `tit_for_tat`, sauf si son adversaire l'a obligé à le punir deux fois de suite, c'est-à-dire si il l'a trahi deux fois de suite. En cela, son comportement est relativement proche de celui de `hard_tf2t`. Le plus étonnant reste donc le choix des cartes jouées durant l'amorce : d'abord une trahison et ensuite une coopération. Si on analyse bien le sens pris par M et O on se rend compte que finalement ce choix est compréhensible. En effet M=2 et O=1, donc le premier choix commandé par la stratégie ne prendra pas en compte le premier coup de l'adversaire. De ce fait chacun a intérêt à trahir au premier coup puisque celui-ci sera sans incidence pour la suite de la confrontation. En revanche, le second coup joué sera le premier pris en compte et dans ce cas il vaut mieux ne pas trahir et tenter la coopération. C'est d'ailleurs ce que font 15 des 20 meilleures stratégies de la classe `mem_2_1`. Par la suite, un grand nombre de stratégies vont donc se retrouver dans le cas de figure où elles auront trahi au premier coup et coopéré au second, alors que leur adversaire aura coopéré au second. 18 des 20 premières stratégies de cette classe se mettent alors à coopérer indéfiniment l'une avec l'autre.

Le fait que ni `bad_bet` ni `gradual` ne fassent de bons résultats dans cette classe n'est alors plus très étonnant. En effet les stratégies de cette classe utilisent un « *dialecte* » particulier qui leur permet de ne pas prendre en compte les trahisons du premier coup. Or `gradual`, aussi bien que `bad_bet`, entament immédiatement leur cycle de punition qui ne leur permet plus de réinstaurer la coopération avec les stratégies de cette classe. Pendant ce temps un grand nombre de stratégies de la classe a pu entamer une coopération fructueuse. Elles se comportent donc bien non seulement face à elles-mêmes mais aussi face à toutes leurs cousines, pendant que `gradual` et `bad_bet` ne se comportent bien avec personne d'autres qu'elles-mêmes.

On peut donc dire que ces classes ont une particuliarité qu'il est difficile d'appréhender pour une stratégie classique. Elles sont trop complexes pour être comprises par des individus qui ne sont pas de leur famille. Le fait qu'elles prennent en compte le comportement de l'adversaire d'une manière moins forte que le leur, peut les faire passer pour irrationnelles. L'adaptation à leur comportement est alors plus difficile, puisque plus masqué que dans le cas où O¿M.

Récapitulatif

Le tableau 5.7 dresse un récapitulatif des résultats des expériences menées. Pour chaque classe complète, il donne le nombre de stratégies de la classe, le nom de la meilleure stratégie, la durée de l'évolution, puis le rang des stratégies `gradual` et `bad_bet` dans des évolutions incluant chacune de ces stratégies et la classe complète, si elles sont classées parmi les 20 meilleures.

Classe	Taille	Meilleure	Générations	gradual	bad_bet
mem_0_1	8	mem_0_1_ccd	33	1	1
mem_0_2	64	mem_0_2_cccdcd	250	5	1
mem_1_1	32	mem_1_1_ccddd	20	2	1
mem_1_2	1 024	mem_1_2_cccdcddcdd	175	10	1
mem_2_1	1 024	mem_2_1_dccdcdcddd	8 647		
memd_0_2	128	memd_0_2_ccccdcd	422		1
memd_1_2	2 048	memd_1_2_ccccdcddcdd	75	15	1
memd_2_1	2 048	memd_2_1_dcccdcdcddd	10 000		
memld_0_3	4 096	memld_0_3_dddccccccdcd	15 000		1
bin_0_1	32	bin_0_1_cddcc	57	1	1
bin_0_2	1 024	bin_0_2_dcddcdcccd	10 842		1
bin_1_1	512	bin_1_1_ccdddcdcc	70	1	11
bind_0_2	2 048	bind_0_2_cccccddccdd	15 000		1
moore_0_1_2	128	moore_0_1_cd_00111	30	3	1
moore_1_1_2	2 048	moore_1_1_cd_001111101	31		1
mealy_0_1_2	1 024	mealy_0_1_c0c0c1c1d1	156	1	1

TAB. 5.7 – *Récapitulatif des expériences sur les classes complètes. Les cases grisées correspondent à des cas où la stratégie n'est pas dans les 20 premières. On voit sur ce tableau la formidable robustesse de* `bad_bet` *qui finit quasiment toujours en tête.*

L'information la plus intéressante de ces simulations est la robustesse de la stratégie `bad_bet` dans ces nombreux environnements différents. Il en ressort que les idées à la base de `bad_bet`, à savoir l'adaptation à l'adversaire et les courtes périodes de mise à jour de l'image que l'on en a, sont des idées fortes. Elles ne permettent cependant pas de bien identifier les raisons du choix des stratégies que `bad_bet` utilise.

La meilleure preuve de la force de `bad_bet` est que ses idées se retrouvent dans le comportement de certaines des meilleures stratégies des classes utilisées dans nos simulations, comme par exemple la meilleure de la classe `mem_1_2` que nous avons décodée page 83.

Pour conclure la figure 5.8 montre une évolution écologique de la classe complète `mem_1_2` dans laquelle sont ajoutées les stratégies `tit_for_tat`, `gradual` et `bad_bet`. Il est intéressant de noter que la hiérarchie de stratégies que nous avons proposée en conclusion du chapitre précédent est strictement respectée.

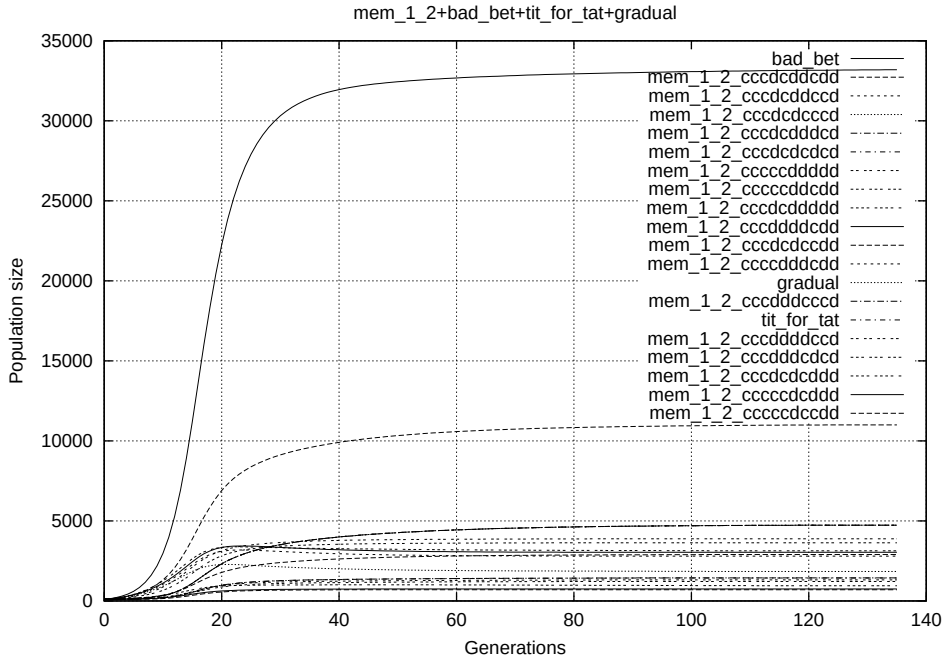


FIG. 5.8 – *Évolutions de la classe complète* `mem_1_2` *avec* `tit_for_tat`, `gradual` *et* `bad_bet`

Chapitre 6

Le dilemme de l'ascenceur

Le dilemme du prisonnier que nous avons étudié dans le reste de la thèse est basé sur le dilemme pur (cf. page 27), c'est-à-dire que, sur le long terme, il favorise la coopération réciproque. Nous présenterons en détail une étude du dilemme *non-pur*, c'est-à-dire en relâchant, et plus exactement en inversant l'inéquation 2.5. Nous appellerons ce nouveau jeu le dilemme de l'ascenceur.

6.1 Un vrai dilemme

Afin de limiter l'étude du dilemme au cas simple dans le cas d'utilisation de stratégies mixtes (cf. page 27) et de favoriser la coopération réciproque sur le long terme dans les versions itérées du dilemme, (cf. 31), le respect de l'inéquation 2.5 est une des hypothèses que nous avons utilisée pour définir ce jeu. Dans cette section, nous allons nous intéresser à l'inversion de cette inéquation et considérer :

$$2R < S + T \tag{6.1}$$

Les résultats présentés ici ont été publiés en détails dans [MD99] et [DM96].

Nous étudierons par exemple l'ensemble des paramètres du dilemme (cf. Tab. 2.2(b), page 22) suivant : $T = 8$, $R = 3$, $P = 1$ et $S = 0$. Il est intéressant de noter qu'un seul paramètre est modifié par rapport au dilemme itéré du prisonnier classique (cf. 41). Nous allons cependant voir que ce léger changement implique des modifications importantes dans les résultats établis sur le dilemme classique.

Nous ne considèrerons que le cas d'un dilemme répété. En effet dans le cas d'un dilemme simple, si on ne considère que des stratégies pures, cette inéquation n'a aucune incidence sur le jeu : seul l'ordre des préférences des issues par les joueurs est important, et cette inéquation ne modifie en rien cet ordre. Le nouveau jeu décrit est donc bel et bien encore un dilemme du prisonnier : l'intérêt collectif est toujours différent de l'intérêt individuel.

Comme pour le dilemme classique, il est aisé :

- de trouver des cycles ($V(\mathtt{A}|\mathtt{B}) > V(\mathtt{B}|\mathtt{A})$, $V(\mathtt{B}|\mathtt{C}) > V(\mathtt{C}|\mathtt{B})$ mais $V(\mathtt{C}|\mathtt{A}) > V(\mathtt{A}|\mathtt{C})$) ;
- de trouver des hiérarchies infinies ($V(\mathtt{A}_1|\mathtt{A}_0) > V(\mathtt{A}_0|\mathtt{A}_1)$, $V(\mathtt{A}_2|\mathtt{A}_1) > V(\mathtt{A}_1|\mathtt{A}_2)$, etc.) ;
- de montrer qu'il n'existe pas de stratégie qui joue de manière optimale contre toutes les autres ;
- de montrer que `all_d` ne perd jamais contre aucune stratégie, mais obtient de faibles scores dans de nombreux cas ;
- de montrer que `tit_for_tat` ne perd jamais plus de T, ici 8, points face à un adversaire quelconque.

Il faut noter qu'il est plus intéressant d'obtenir alternativement les issues [C,D], [D,C], que [C,C], [C,C]. Dans le premier cas la totalité des points distribués au joueur A est $S + T$, et dans le second

$R + R = 2R$. Donc il existe désormais deux niveaux de coopération :

- le niveau de base, qui correspond à une sorte de pacte de non agression, utilise une répétition de coups [C,C]. En moyenne une telle coopération rapporte $(R + R)/2 = R$, soit 3, points.
- un méta niveau, qui correspond à une sorte de pacte de réciprocité, qui utilise une alternance de coups [C,D] et [D,C]. En moyenne une coopération de ce type rapporte $(T + S)/2$, soit 4, points.

La méta-coopération est donc clairement plus intéressante que la coopération grâce au respect de l'inéquation 6.1.

Pour réussir une méta-coopération, les joueurs doivent jouer en *opposition de phase*, c'est-à-dire que quand l'un joue C, l'autre joue D, et réciproquement. Cependant cette opposition de phase est difficile à obtenir puisqu'elle nécessite une entente et une coordination entre les joueurs. La mise au point de cette entente peut être risquée, notamment pour le premier joueur qui joue C.

Nous nommons ce jeu le dilemme itéré de l'ascenceur. En effet on peut considérer que ce jeu modélise des situations où un des agents laisse un autre l'exploiter en espérant que l'autre lui *renverra l'ascenceur* en se laissant exploiter plus tard.

De plus, il existe de nombreuses instances de méta-coopérations. En effet l'exemple le plus simple correspond à une alternance de coups sur une période de 1, c'est-à-dire un match de la forme [CDCDCD,DCDCDC], correspondant à l'utilisation de la stratégie (CD)$*$ pour le joueur A, et de la stratégie (DC)$*$ pour le joueur B. On peut également utiliser des alternances avec des périodes plus importantes : un match de la forme [CCCDDD,DDDCCC], correspondant à l'utilisation de la stratégie (CCCDDD)$*$ par A et de la stratégie (DDDCCC)$*$ par B est aussi une méta-coopération. En moyenne, les deux matchs rapportent autant à chacun des joueurs. Ces niveaux de coopération demandent cependant plus de coordination et de confiance en l'adversaire.

En fait désormais dans le dilemme de l'ascenceur il peut exister de nombreux états de coopérations différents, mais atteindre ces états est beaucoup plus difficile que dans le dilemme pur. Rappelons que les conditions du jeu sont exactement les mêmes que pour le dilemme pur, et que par conséquent il est impossible aux joueurs de s'entendre avant la partie pour choisir un niveau de coopération, ni même au cours du jeu.

Le seul moyen de communiquer est d'utiliser l'historique des coups joués dans la partie [30], comme c'est le cas dans le dilemme pur et itéré du prisonnier. Il est à noter que la communication est moins coûteuse que dans le cas du dilemme pur puisqu'un coup [C,D] n'est pas forcément une mauvaise chose en moyenne, ce qui est faux dans le dilemme pur.

De nombreuses situations de conflits entre agents humains ou artificiels sont modélisées par le dilemme de l'ascenceur, notamment celles où un objet est donné périodiquement aux agents avec la limitation qu'un seul des deux puisse l'acquérir. Un grand nombre d'exemples d'applications du modèle est donné dans [MD99], nous ne les reprendrons pas en détail ici. On peut cependant citer le cas des sessions de recrutements, dans les grandes entreprises ou les instutions, des ventes aux enchères, et plus généralement le cas de l'accès périodique à un objet ou un service indivisible.

On peut enfin souligner que ce nouveau jeu comporte dans sa définition quelques points qui le différencient très nettement du dilemme pur :

- Jouer D peut être interprété de deux manières :
  1. refuser de coopérer
  2. méta-coopérer

  Il est donc de la responsabilité des joueurs d'interpréter les actions des adversaires d'une manière ou d'une autre. On a donc là une difficulté nouvelle par rapport au dilemme pur, quant aux croyances qu'un joueur a sur les motivations de son adversaire.

30. On rappelle que les joueurs peuvent modifier leur croyances sur le comportement de l'adversaire au cours de la partie (cf. page 30)

- Ne pas vouloir méta-coopérer, mais uniquement coopérer ne doit pas être considéré comme une trahison et un refus de collaboration. On a ici apparition de niveaux de comportement beaucoup plus subtiles qu'au dilemme pur. On pourrait faire un classement de ceux-ci en disant qu'il existe trois niveaux de comportement :
  1. ne pas coopérer du tout
  2. coopérer prudemment
  3. coopérer avec risque

  Les deux derniers comportements sont des modèles de coopération, et même si un des deux correspond à une coopération plus fructueuse que l'autre, un *bon* comportement ne doit pas prendre en compte un seul des niveaux de coopération.
- La réactivité, comme au dilemme pur, est importante, mais une fois de plus, elle correspond à deux aspects différents :
  - punition d'une agression
  - adaptation à un niveau de méta-coopération de période fixe (2, 3 etc.)

  Les réactions adaptatives sont beaucoup plus risquées, dans le sens où elles demandent des périodes d'acceptation de la trahison de l'adversaire plus longues.

Dans la perspective d'évolutions écologiques il faut qu'un joueur se comporte bien face à un joueur utilisant le même comportement que lui. Cette propriété est aisément définissable dans le dilemme pur itéré, puisqu'il suffit de de ne jamais être le premier à jouer `D`. Nous allons voir que pour le dilemme de l'ascenceur la chose est plus complexe.

6.2 Évolution écologique en environnement homogène

Nous allons donc maintenant nous intéresser à la recherche de stratégies obtenant le meilleur score possible dans un environnement homogène, c'est-à-dire celles qui sont capables de bien se comporter face à elle-même.

Définition 6.1 *Une stratégie est* **rationnelle** *(resp.* **asymptotiquement rationnelle***) si lorsqu'elle joue contre elle-même elle obtient le meilleur score possible pour toutes les parties de longueur n (resp. asymptotiquement le meilleur score possible).*

Notons $\mathrm{V}_n(\mathrm{A})$ le score obtenu par une stratégie A jouant contre elle même pendant n coups (quand la stratégie est probabiliste $\mathrm{V}_n(\mathrm{A})$ correspond à l'espérance de gain sur n coups).

Par définition, une stratégie est dite *rationnelle* si

$$\forall n : V_n(A) = \max\{V_n(X); X \text{ est une stratégie}\}$$

Par définition, une stratégie est dite *asymptotiquement rationnelle* si :

$$\lim_{n\to\infty} \left[\frac{V_n(A)}{\max\{V_n(X); X \text{ est une stratégie}\}} \right] = 1$$

Dans une évolution écologique avec une population homogène une stratégie rationnelle obtient le meilleur gain possible. Une telle stratégie a donc un avantage quand elle rencontre d'autres stratégies, notamment dans une évolution écologique dans laquelle sa population de départ est suffisamment importante.

Pour être efficace lors de rencontres avec d'autres individus le score total ne doit pas obligatoirement être distribué de manière équitable entre tous les individus utilisant la même stratégie. C'est pourquoi nous introduisons la notion de stratégies collectivement rationnelles et collectivement asymptotiquement rationnelles.

Définition 6.2 *Une stratégie est* **collectivement rationnelle** *(resp.* **collectivement asymptotiquement rationnelle***) si quand elle joue contre elle même le score global collecté par les deux est*

le meilleur possible pour toutes les parties de longueur n (resp. asymptotiquement le meilleur score possible).

Formellement on note V'$_n$(A) le score global obtenu par deux copies de la stratégie A jouant l'une contre l'autre pendant n coups (quand la stratégie est probabiliste V'$_n$(A) est l'espérance du gain total sur n coups).

Par définition une stratégie est dite *collectivement rationnelle* si :

$$\forall n : V'_n(A) = \max\{V'_n(X); X \text{ est une stratégie}\}$$

Par définition, une stratégie est dite *asymptotiquement collectivement rationnelle* si :

$$\lim_{n\to\infty} \left[\frac{V'_n(A)}{\max\{V'_n(X); X \text{ est une stratégie}\}} \right] = 1$$

6.3 Le cas du dilemme classique

Dans le dilemme classique, c'est-à-dire avec $T = 5$, $R = 3$, $P = 1$, et $S = 0$, les stratégies rationnelles sont celles qui sont gentilles, *i.e.* ne prennent jamais l'initiative de la première trahison. En effet pour obtenir le meilleur score possible, une stratégie doit toujours jouer `D`, c'est-à-dire un coup [`C`,`C`], qui rapporte collectivement $2R$ (6) points à chaque coup. Un seul changement dans cette suite de cartes `C`, produira une baisse du score collectif.

Les stratégies asymptotiquement rationnelles sont celles qui sont capables d'obtenir, en moyenne, R (3) points par coup quand elles jouent contre elles-mêmes. Elles peuvent trahir de temps en temps, de manière déterministe ou non, mais le nombre de trahisons doit varier de ∞ vers 0 (par exemple trahir avec une probabilité $1/n$ au coup n).

Dans le dilemme classique, les notions de rationalité collective ou de rationalité asymptotique collective n'ont pas beaucoup d'intérêt. En effet, il n'y a qu'une seule manière d'obtenir un meilleur score et c'est de se retrouver dans une configuration [`C`,`C`]. Les stratégies collectivement rationnelles sont donc rationnelles, et les stratégies collectivement asymptotiquement rationnelles sont asymptotiquement rationnelles.

Dans le dilemme de l'ascenceur, la situation est toute autre, car de temps en temps une stratégie va devoir se sacrifier pour une de ses sœurs. Certaines stratégies collectivement rationnelles ne sont pas rationnelles.

6.4 Le cas du dilemme de l'ascenceur

Nous présentons ici quelques résultats mathématiques sur le dilemme de l'ascenceur.

Théorème 6.1 *Si les deux inégalités suivantes sont respectées, une stratégie déterministe n'est jamais rationnelle ni asymptotiquement rationnelle :*

$$T > R > P > S$$
$$S + T > 2R$$

Preuve.
Quand une stratégie déterministe joue contre elle-même, elle ne peut jamais se retrouver dans une configuration [`C`,`D`] ou [`D`,`C`]. Dans le meilleur des cas elle peut donc gagner R (3) points à chaque coup.
Nous allons voir qu'il existe des stratégies probabilistes qui obtiennent en moyenne $(T + S)/2$ (4) points par coups. □

On appelle un coup en phase un coup [C,C] ou [D,D], et un coup déphasé un coup [C,D] ou un coup [D,C].

Théorème 6.2 (Caractérisation rationnelle) *Une stratégie est rationnelle si et seulement si :*

- *au premier coup et tant que le dernier coup n'est pas en déphasage, elle joue la carte* C *avec une probabilité de* $opti = 0{,}56696$ *et la carte* D *avec une probabilité de* $1 - opti$ *;*
- *après le premier coup déphasé, elle utilise une règle qui pour chaque histoire du jeu possible contre elle-même assure que celle qui a joué* D *au premier coup déphasé, joue la carte opposée de celle qui a joué* C *au premier coup déphasé. Ceci assure qu'après le déphasage une stratégie jouant contre elle-même est déterministe.*

Preuve.
Le second point est trivial.
opti est la solution d'un polynôme utilisé pour minimiser la durée de la période de recherche d'un coup déphasé pour une stratégie jouant contre elle-même. La méthode la plus simple pour obtenir ce déphasage serait de trahir et de coopérer avec la même probabilité de 0,5*, le nombre moyen de coup en phase étant alors de 1.*
Avec cette méthode, la stratégie perd, en moyenne, plus de points à chaque coup que si elle jouait C *avec une probabilité de* 0,75 *et* D *avec une probabilité de* 0,25 *par exemple. En effet dans ce cas, les coups [*C,C*], rapportant R (3) points sont plus fréquents que les coups [*D,D*], ne rapportant que P (1) point. Il n'est donc pas évident que la meilleure méthode pour obtenir un déphasage soit de jouer* C *et* D *avec une probabilité équivalente de* 0,5*.*
Plus généralement, étudions le cas où C *est joué avec une probabilité* p *et* D *avec une probabilité* $1-p$*. Pendant la période recherche de déphasage en moyenne la stratégie gagne donc* $Rp + P(1-p)$ *soit* $p(R-P)+P$ *points par coups.*
Le calcul de la durée moyenne de cette période est donc :

durée	**coups**	**probabilité**
0	*[D,C] ou [C,D]*	$2p(1-p)$
1	*([D,D] ou [C,C]) suivi de ([D,C] or [C,D])*	$2p(1-p)(p^2+(1-p)^2)$
2	*([D,D] ou [C,C]) deux fois suivi de ([D,C] ou [C,D])*	$2p(1-p)(p^2+(1-p)^2)^2$
$\vdots$	$\vdots$	$\vdots$

L'espérance de la durée de cette période, EL, est alors :

$$\begin{aligned} EL &= 2p(1-p)\left[(p^2+(1-p)^2) + 2(p^2+(1-p)^2)^2 \right. \\ &\quad \left. + 3(p^2+(1-p)^2)^3 + 4(p^2+(1-p)^2)^4 + \cdots\right] \end{aligned}$$

Or

$$\begin{aligned} & X + 2X^2 + 3X^3 + 4X^4 + \cdots = \\ &= X[1 + 2X + 3X^2 + 4X^3 + \cdots] \\ &= X[1 + X + X^2 + X^3 + \cdots]' \\ &= X\left[\frac{1}{(1-X)}\right]' \\ &= \frac{X}{(1-X)^2} \end{aligned}$$

$$\begin{aligned} \text{D'où, } EL &= \frac{2p(1-p)\left(p^2+(1-p)^2\right)}{\left[1-(p^2+(1-p)^2)\right]^2} \\ &= \frac{2p(1-p)\left(p^2+(1-p)^2\right)}{\left(2p(1-p)\right)^2} \end{aligned}$$

$$= \frac{\left(p^2 + (1-p)^2\right)}{(2p\,(1-p))}$$
$$= \frac{1 - 2p + 2p^2}{2p(1-p)}$$

La perte L (comparée à un jeu immédiatement déphasé) est :

$$L = \left[\frac{(1-2p+2p^2)}{2p(1-p)}\right]\left[\frac{(T+S)}{2} - [p(R-P)+P]\right]$$
$$= \frac{T+S}{2} - \frac{[p(R-P)+P](1-2p+2p^2)}{2p(1-p)}$$

Avec nos paramètres, on obtient :

$$L = \frac{(2p-3)(1-2p+2p^2)}{2p(1-p)}$$

Pour $p = 0{,}5$ *on trouve* $L = 2$*; pour* $p = 0{,}7$ *on trouve* $L = 2.209$*; pour* $p = 0{,}9$ *on trouve* $L = 5.46$*; pour* $p = 0{,}4$ *on trouve* $L = 2.38$*; pour* $p = 0{,}6$ *on trouve* $L = 1.95$*; pour* $p = 0{,}55$ *on trouve* $L = 1.938$*; pour* $p = 0{,}65$ *on trouve* $L = 2.03$
La perte minimale est obtenue pour $p = 0{,}5669640801\ldots$.
La période de recherche d'un dépahasage la moins coûteuse est obtenue lorsque `C` *est joué avec une probabilité* $opti = 0{,}5669640801\ldots$ *et que* `D` *est joué avec une probabilité* $1 - opti$. □

Le coût de cette recherche est :

$$-(1-2p+2p^2)(p(R-P)+P-(T+S)/2)/[2p(1-p)]$$

ce qui donne avec nos paramètres $T = 8$, $R = 3$, $P = 1$, $S = 0$:

$$-(2p-3)(1-2p+2p^2)/2p(1-p)$$

On peut noter que ce gain obtenu en utilisant opti = 0,56696 au lieu de opti = 0,5 est de moins de 1/10 point. Par simplicité nous utiliserons donc souvent des périodes de recherche avec $p = 0{,}5$.

6.4.1 Quelques exemples

La stratégie rationnelle la plus simple est appelée `reason` :

`reason`

- je joue `C` avec une probabilité de 0,56696 et `D` avec une probabilité de 0,43304 au premier coup et tant que le dernier coup est en phase ;
- ensuite
  - si le premier coup déphasé a été [`C`,`D`], je joue (`DC`)$*$
  - si le premier coup déphasé a été [`D`,`C`], je joue (`CD`)$*$

La stratégie suivante, appelée `naive_reason`, est collectivement rationnelle mais n'est pas rationnelle. Une population homogène d'individus jouant `naive_reason` obtient donc globalement le

meilleur score possible, mais les scores ne sont pas distribués équitablement parmi ces individus.

`naive-reason`

- je joue `C` avec une probabilité 0,56696 et `D` avec une probabilité de 0,43304 au premier coup et tant que le dernier coup est en phase ;
- ensuite
  - si le premier coup déphasé a été [`C`,`D`], je joue (`C`)$*$
  - si le premier coup déphasé a été [`D`,`C`], je joue (`D`)$*$

On peut traduire le comportement de cette stratégie par : « *Si au premier désaccord je me suis fait exploité, alors je considère être le perdant et je me résigne. Si je gagne au premier désaccord, alors je dois toujours gagner* ».

De telles règles utilisées par des individus d'une même espèce sont à même d'établir des hiérarchies. Ce genre de règles respecte la rationalité collective, mais n'assure pas l'égalité entre les individus. On peut également noter que cette stratégie est plus simple que la stratégie `reason` [31].

Les deux stratégies suivantes essaient d'améliorer `reason` en étant plus gentille. L'idée est de ne pas « *énerver* » les stratégies adversaires trop susceptibles.

`gentle-reason`

Je joue comme `reason` à l'exception du fait que la probabilité de jouer la carte `D` pendant les trois premiers coups est nulle.

`reason-[a,1-a]`

(a est une paramètre entre 0 et 1)
Je joue comme `reason` à l'exception du fait que durant la période de recherche de déphasage, je joue la carte `C` avec la probabilité a et `D` avec la probabilité $1-a$.

Ces deux stratégies sont asymptotiquement rationnelles parce qu'elles perdent peu de points lors de la recherche du déphasage par rapport à ce qu'elles peuvent espérer de mieux.

Bien que similaire à `reason`, la stratégie suivante n'est ni rationnelle, ni collectivement rationnelle, ni asymptotiquement rationnelle, ni collectivement asymptotiquement rationnelle. En fait elle se satisfait trop rapidement d'un niveau de coopération simple, et donc d'une moyenne de R points par coups.

`coop-reason`

je joue comme la stratégie `all_c` jusqu'à la première trahison de l'adversaire de mon adversaire, après quoi je joue la stratégie `reason`.

6.4.2 À propos du déterminisme

Dans le théorème 6.2, nous avons noté « *. . . pour chaque histoire du jeu possible contre elle même. . .* ». En effet il est uniquement important de jouer en opposition de phase contre soi même.

31. On peut peut-être trouver là une explication au fait que les sociétés despotiques sont apparues avant les démocraties.

Considérons par exemple la stratégie suivante :

`reason-careful`

- je joue `C` avec une probabilité 0,56696 et `D` avec une probabilité 0,43304 au premier coup et jusqu'à ce que le dernier coup soit déphasé ;
- ensuite
  - si le premier coup déphasé a été [`C`,`D`] je joue (`DC`)$*$ sauf si mon adversaire a joué trois fois de suite `D` depuis le début du déphasage auquel cas je joue (`D`)$*$;
  - si le premier coup déphasé a été [`D`,`C`] je joue (`CD`)$*$ sauf si mon adversaire a joué trois fois de suite `D` depuis le début du déphasage auquel cas je joue (`D`)$*$.

Cette stratégie est rationnelle car ses règles respectent le théorème 6.2. La stratégie suivante est également rationnelle :

`reason-tit_for_tat`

- je joue `C` avec une probabilité 0,56696 et `D` avec une probabilité 0,43304) au premier coup et jusqu'à ce que le dernier coup soit déphasé ;
- ensuite je joue la stratégie `tit_for_tat`.

La stratégie suivante est une généralisation de celles que nous venons de voir. Avec a proche de 1, elles sont moins agressives au début des parties.

`reason-[a,1-a]-tit_for_tat`

- je joue `C` avec une probabilité a et IPD avec une probabilité $1-a$ au premier coup et jusqu'à ce que le dernier coup soit en opposition de phase ;
- ensuite je joue la stratégie `tit_for_tat`.

6.4.3 Remarques concernant la période de recherche de déphasage

La valeur 0,56696 a été calculée sous l'hypothèse d'homogénéité de la population, c'est-à-dire que tous les individus de la poulation utilisent la même stratégie. Il est clair que cette valeur n'est pas optimale dans une population hétérogène.

Dans un panel hétérogène, une stratégie a tout intérêt à être celle qui trahit lors du premier coup déphasé. En effet si la partie dure un nombre impair de coups, alors elle aura un avantage sur son adversaire. Cet avantage peut être évalué à $(S+T)/2$: si la partie dure un nombre pair de coups il vaut S, sinon T. Cet avantage est très important dans le cas de partie très courte pour lesquelles les scores sont faibles.

Il est évident qu'il n'y a pas de valeur optimale dans un tel cas, puisque celle-ci dépend du panel de départ. Les simulations que nous avons effectuées sur diverses stratégies `reason-[a,1-a]` ont montré les résultats suivants :

- pour une évolution écologique impliquant 2 types de stratégies : $a = 0{,}5$ l'emporte sur $a = 0{,}567$;
- pour une évolution écologique avec 5 types de stratégies l'ordre de préférence est : $a = 0{,}5$; $a = 0{,}433$; $a = 0{,}567$; $a = 0{,}1$; $a = 0{,}9$;

Ces résultats ne permettent pas de déduire de conclusions générales sinon que les choix de a proche de 0 ou de 1 ne sont pas de bons choix. Il est de plus évident que ces résultats sont très sensibles aux qualités du générateur aléatoire utilisé pour les simulations, et peuvent fluctuer selon les conditions expérimentales (nous avons ici fait la moyenne de 1000 tournois).

Afin d'avoir un bon comportement, il est possible d'étudier le comportement de son adversaire afin de s'y adapter. Par exemple si l'adversaire trahit souvent consécutivement, il est certainement en train d'essayer d'exploiter le joueur. Dans un tel cas, l'intérêt du joueur est certainement d'essayer de recommencer à chercher un déphasage. La stratégie suivante se base sur cette idée, en faisant un test après chaque suite de trois trahisons.

`iterated-reason`

- je joue `C` avec une probabilité 0,56696 et `D` avec une probabilité 0,43304 au premier coup et jusqu'à ce que le dernier coup soit déphasé ;
- ensuite
  - si le premier coup déphasé a été [C,D] je joue (DC)$*$ sauf si mon adversaire a joué trois fois de suite `D` auquel cas je recherche à obtenir un déphasage comme au début de la partie ;
  - si le premier coup déphasé a été [D,C] je joue (CD)$*$ sauf si mon adversaire a joué trois fois de suite `D` auquel cas je recherche à obtenir un déphasage comme au début de la partie.

6.4.4 Remarques à propos de la complexité des stratégies

L'utilisation du premier coup en opposition de phase dans le théorème 6.2 implique que les stratégies rationnelles doivent être en mesure de conserver en mémoire le premier coup déphasé. De ce fait, les stratégies rationnelles en plus de devoir être probabilistes doivent posséder une mémoire. En terme informatique, on peut donc dire que leur complexité n'est pas négligeable.

Plus largement on peut considérer le dilemme de l'ascenceur comme une version du dilemme du prisonnier dans laquelle la communication est moins coûteuse. Les adversaires peuvent donc au travers de l'histoire du jeu s'entendre sur une méthode de méta-coopération. Le revers de la médaille est que les joueurs doivent pouvoir se souvenir tout au long de la partie de cet arrangement.

Bien se comporter face à soi-même, c'est-à-dire avoir une première forme de rationalité, implique donc nécessairement un certain niveau de complexité. On peut une fois de plus trouver en cela un argument contre la qualité de la simplicité et de la clarté dans les comportements pour la coopération. Coopérer n'est pas un mécanisme simple, pas plus au dilemme de l'ascenceur qu'au dilemme du prisonnier.

Nous présentons maintenant certaines des nombreuses expériences que nous avons menées et qui confirment les résultats et idées que nous venons d'exposer.

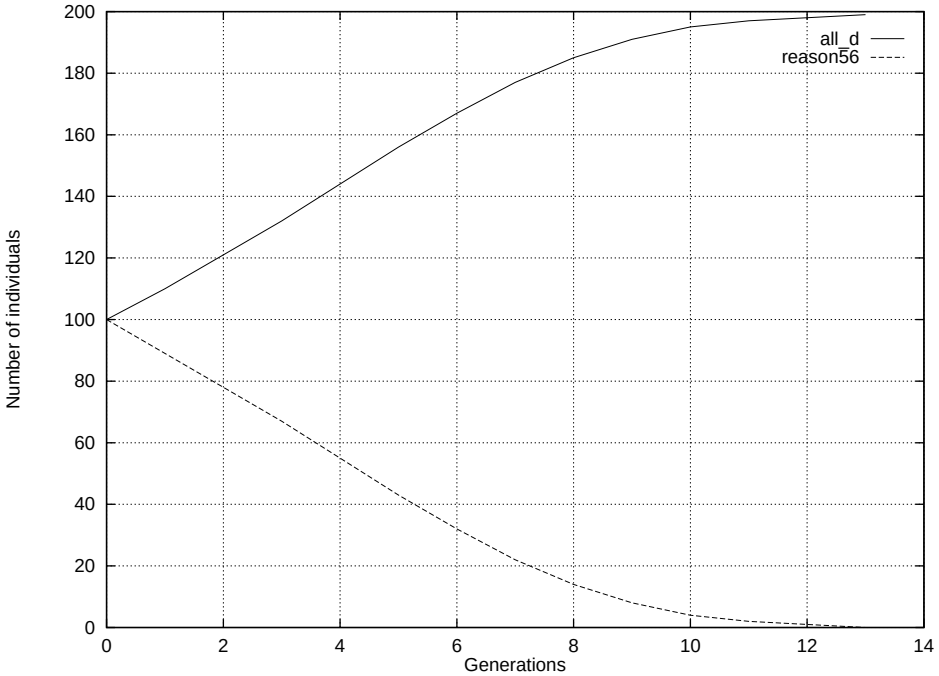
6.5 Études expérimentales

6.5.1 `all_d` vs `reason`

La stratégie `reason` se comporte très bien face à elle-même, mais n'est pas du tout réactive. De ce fait, elle peut être exploitée de manière très simple, par exemple par la stratégie `all_d`. Voyons son comportement dans une partie :

`all_d` contre `reason` donne

$$[D,D][D,D]\cdots[D,D][D,C] \ + \ [D,D][D,C][D,D][D,C][D,D]\cdots$$




```
Score list after tournament
1                all_d  =   5498
2              reason50 =   4498
```

FIG. 6.1 – `all_d` *vs.* `reason`

Après que le premier coup déphasé soit atteint (juste avant le +), ce qui ne prend pas beaucoup de temps, la confrontation continue par l'exploitation de `reason` : en moyenne `reason` gagne $P + S$ points et `all_d` $P + T$. Dans une confrontation sur 1000 coups par exemple, `all_d` va obtenir $1000T$ face à elle-même et $1000(T + S)$ face à `reason` alors que `reason`, quant à elle, va obtenir $500(T + S)$ face à elle-même et $1000P$ face à `all_d`. `all_d` va donc rapidement pouvoir prendre l'avantage sur `reason`. Une telle confrontation est représentée sur la figure 6.1.

6.5.2 `all_d` vs `reason-tit_for_tat`

Cette simulation représentée sur la figure 6.2 montre que la modification de `reason` par ajout des idées de `tit_for_tat` améliore grandement son comportement puisque désormais elle peut battre `all_d`.

6.5.3 `tit_for_tat` vs `reason`

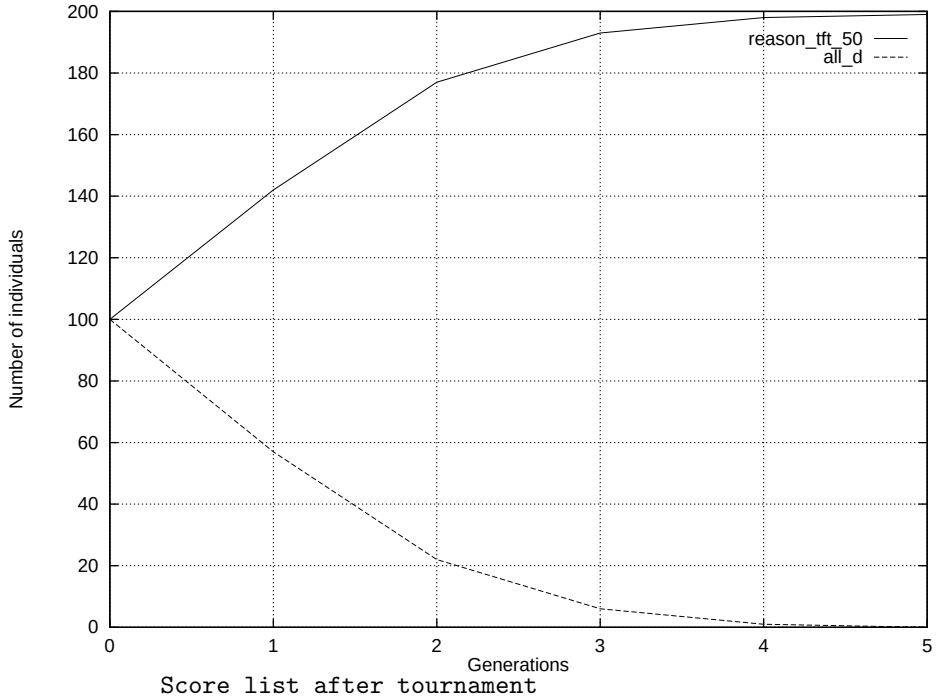
Une fois de plus, la réputation de `tit_for_tat` est malmenée dans cette évolution.

Dans le dilemme de l'ascenceur, les stratégies déterministes ne peuvent être de bonnes stratégies. `tit_for_tat` est déterministe, mais elle est cependant capable d'établir une coopération de haut niveau avec son adversaire en utilisant des périodes de deux coups. En fait elle se comporte très bien face à `reason`, en gagnant en moyenne $(T + S)/2$ points, face à elle, mais se contente d'une coopération simple face à elle-même, soit $2R$ points. C'est cette simplicité qui lui est alors fatale en évolution écologique comme l'illustre la figure 6.3.

`tit_for_tat` contre `reason` donne

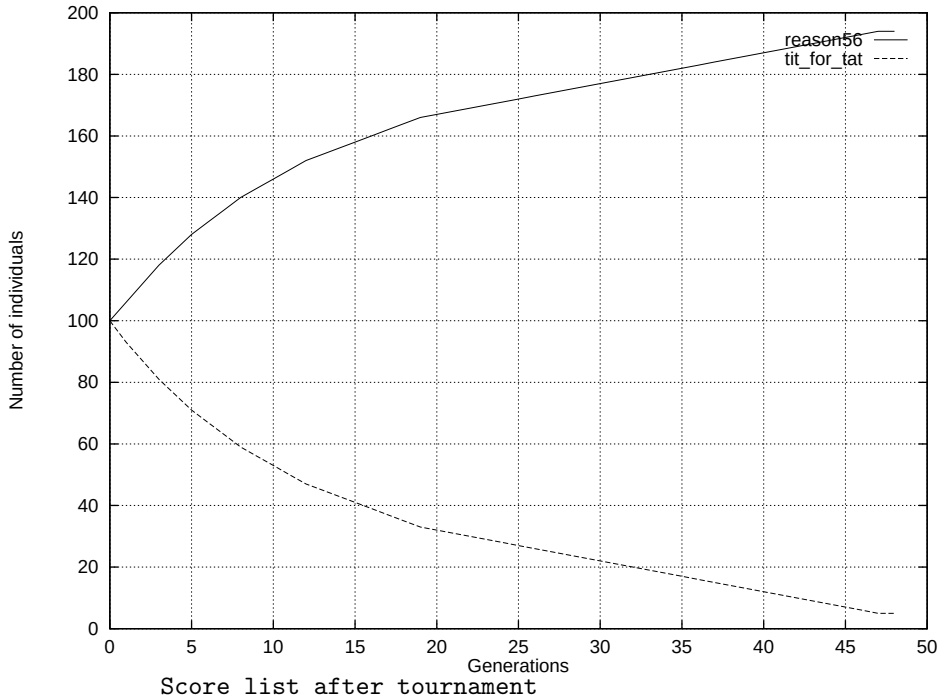
[C,C][C,C]···[C,C][C,D]+ [C,D][D,C][C,D][D,C][C,D]···

Dans une partie de 1000 coups `tit_for_tat` contre elle-même ne gagne que $2000R$ points alors que `reason` gagne $1000(S + T)$ contre elle-même. En moyenne, `tit_for_tat` et `reason` font le même score lorsqu'elles jouent l'une contre l'autre, à savoir $500(S + T)$.




```
Score list after tournament
1         reason_tft_50 =   4997
2                 all_d =   2007
```

FIG. 6.2 – `all_d` *vs.* `reason-tit_for_tat`




```
Score list after tournament
1              reason56 =   7996
2           tit_for_tat =   6997
```

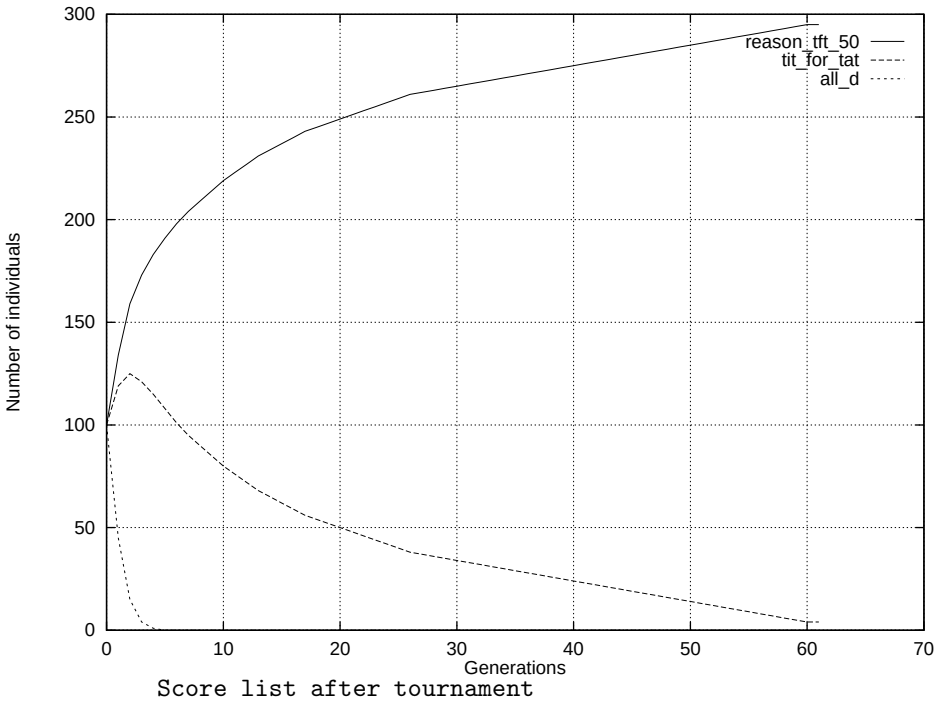
FIG. 6.3 – `tit_for_tat` *vs.* `reason`

6.5.4 `reason-tit_for_tat`, `tit_for_tat` et `all_d`

Les parties rencontrées dans ce genre d'évolution sont :

- `reason-tit_for_tat` contre `tit_for_tat` qui donne :
  `[C,C][C,C]···[C,C][C,D]+ [D,C][C,D][D,C]···` ce qui fait une moyenne de $(T+S)/2$ (4) par coup ;
- `reason-tit_for_tat` contre `all_d` qui donne
  `[D,D][D,D]···[D,D][C,D]+ [D,D][D,D][D,D][D,D]···` ce qui fait en moyenne P (1) par coup et par joueur ;
- `tit_for_tat` contre `all_d` qui donne `[C,D][D,D][D,D][D,D]···` ce qui fait en moyenne P (1) par coup et par joueur ;
- `all_d` contre elle-même gagne en moyenne gagne P (1) point par coup ;
- `tit_for_tat` contre elle-même en moyenne gagne R (3) points par coup ;
- `reason-tit_for_tat` contre elle-même en moyenne gagne $(T+S)/2$ points par coup.

Dans une évolution écologique, moins une stratégie se comporte bien face à elle-même plus vite elle disparaît. Ceci est encore plus visible dans le dilemme de l'ascenceur comme le montre la figure 6.4, dans laquelle on voit disparaître successivement `all_d` puis `tit_for_tat`.




```
Score list after tournament
1         reason_tft_50 =   8997
2           tit_for_tat =   7996
3                all_d =   3014
```

FIG. 6.4 – `reason-tit_for_tat`, `tit_for_tat`, *et* `all_d`

On peut noter que `tit_for_tat` profite de la faiblesse de `all_d` face à `reason_tft_50` jusqu'à sa disparition complète après quoi le sort en est jeté.

6.5.5 `reason-tit_for_tat` contre 10 stratégies de base.

Afin d'évaluer de manière simple si `reason-tit_for_tat` est une *bonne* stratégie, elle est comparée à 10 stratégies classiques du dilemme du prisonnier [32]. Comme on peut le voir sur la figure 6.5, `tit_for_tat` s'en sort mieux que `reason_tit_for_tat`, mais que cela ne dure pas très longtemps,

32. Rappelons que le dilemme de l'ascenceur est à la base un dilemme du prisonnier.

juste le temps de voir disparaître, ou en tout cas suffisamment faiblir les stratégies se comportant mal face à elles-mêmes.

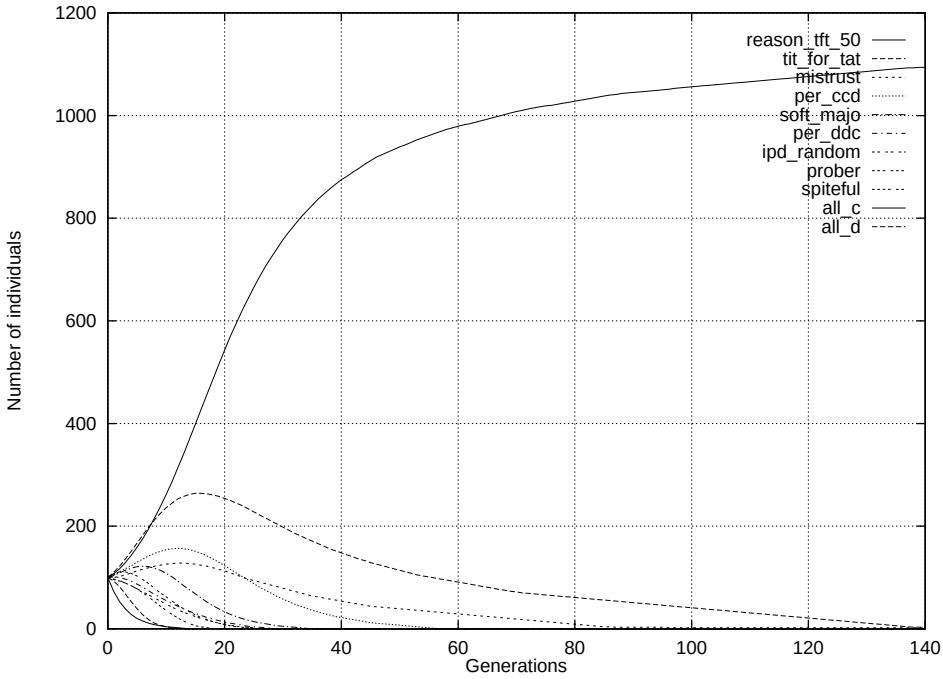


FIG. 6.5 – `reason-tit_for_tat` *contre 10 stratégies de base*

6.5.6 Sensibilité aux paramètres

La courbe de la figure 6.6 indique l'influence de la variation du paramètre T sur la dynamique des populations.

Les expériences faites ici prennent en compte dans un premier temps un panel de quatre stratégies simples, pour lesquelles des évolutions sont menées avec différentes valeurs de T, puis les mêmes expériences sont reconduites en ajoutant `reason` dans le panel. Les valeurs de T varient de $T = 5$, cas du dilemme classique, à $T = 8$, cas du dilemme de l'ascenceur. On constate qu'être probabiliste ne suffit pas pour être bon au dilemme de l'ascenceur.

Être probabiliste ne suffit pas pour atteindre un méta-niveau de coopération, un minimum de structure, de mémoire et donc de complexité est nécessaire pour bien appréhender une communication efficace avec son adversaire.

6.6 Conclusion

Nous avons donc étudié la version **non pure** du dilemme du prisonnier dans sa version répétée. Cette version est un dilemme du prisonnier particulier. Nous avons montré que dans ce modèle la coopération peut intervenir à deux niveaux différents. La communication via l'histoire du jeu est donc non seulement facilitée, mais aussi indispensable, afin d'obtenir ce méta-niveau.

De ce fait, nous avons montré que seules les stratégies probabilistes, c'est-à-dire non déterministes ont une chance dans ce genre de jeu. L'analyse du jeu est rendue beaucoup plus complexe encore que dans le dilemme itéré classique. Cependant, les stratégies *bonnes* dans ce modèle sont plus intéressantes car elles permettent d'étudier tout un spectre de situations de choix de résolutions de conflits différents, un peu comme si les joueurs du dilemme du prisonnier classique découvraient la parole.

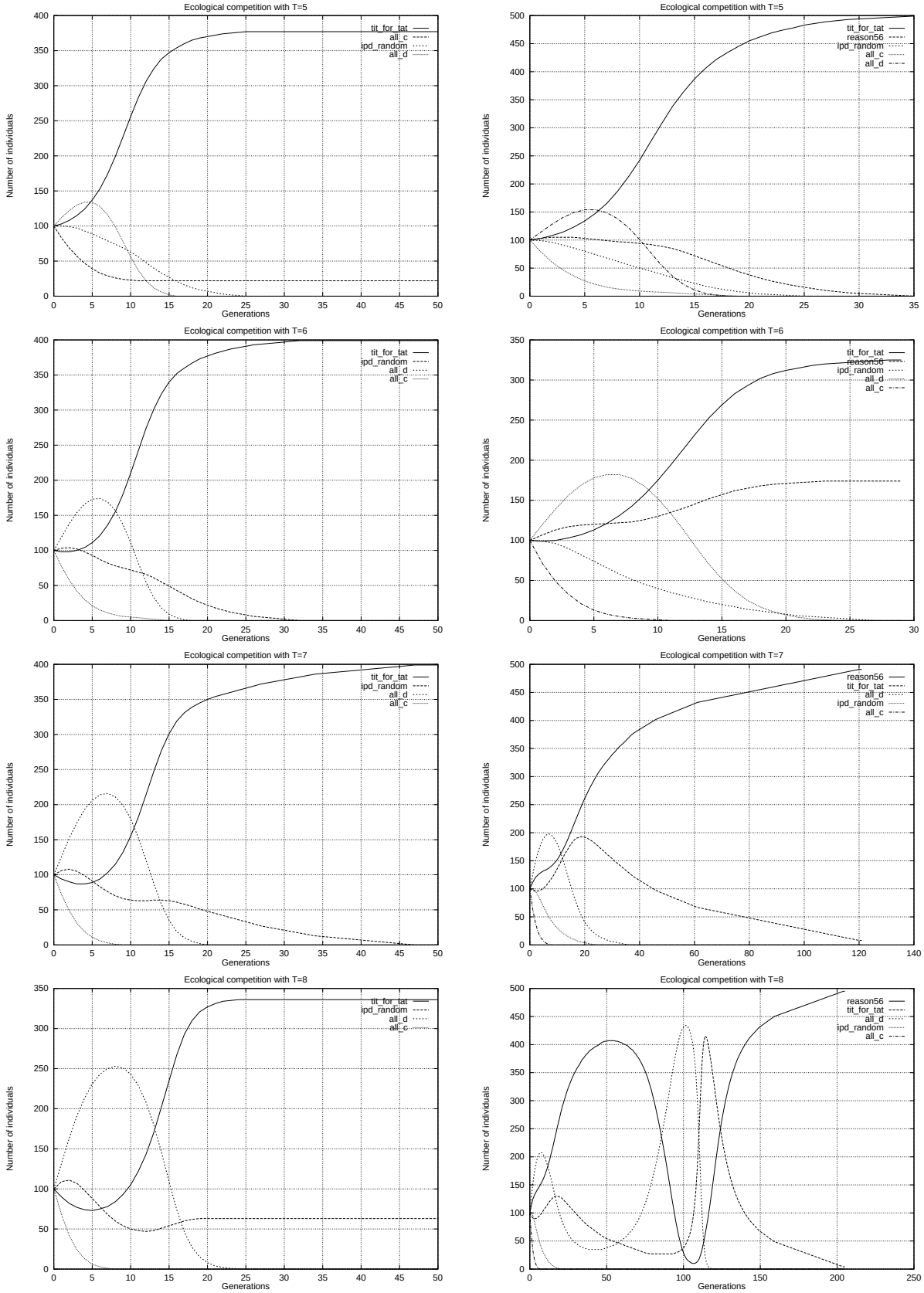


FIG. 6.6 – *Sensibilité aux variations de T*

Conclusion

Le travail de cette thèse est lié aux simulations informatiques d'agents utilisant le dilemme itéré du prisonnier. Il est directement inspiré par les travaux d'Axelrod. Les résultats qui en découlent sont cependant légèrement différents.

Dans un premier temps nous avons souligné les différences méthodologiques, et surtout philosophiques de l'approche des horizons infinis. Axelrod considère que la probabilité d'arrêt de vie d'un agent doit être prise directement en compte dans le calcul de l'évaluation d'un comportement, alors que nous considérons, en partie pour des raisons pragmatiques, que ceci doit être simplement pris en compte par les conditions de simulations. Nous avons montré le peu de différences que cela implique sur la nature des résultats.

Nous avons, par ailleurs, montré la complexité de l'étude de la coopération en étudiant le cas de l'améliorabilité des panels de stratégies. Cette complexité est mise en avant par la construction d'un cas où une infinité de panels sont améliorables, et par la démonstration que dans le cas général ceci n'est pas vrai. Nous avons donc montré que les environnements sont de types très différents, et que cette nature joue forcément un grand rôle dans l'évaluation de la qualité d'un comportement coopératif.

Le principe de base de nos simulations est avant tout le déterminisme, et la discrétisation des calculs. Ces deux limitations qui semblent, a priori, éloigner nos simulations des cas théoriques de la biologie, se montrent cependant peu contraignantes. Elles n'empêchent pas de confirmer certains résultats classiques, comme, la présence d'oscillations dans la dynamique des populations, [NS89]. Nos simulations ont permis d'étendre ces mêmes résultats, et de découvrir certaines propriétés nouvelles, comme, dans certains cas rares, la forte sensibilité de l'évolution des comportements aux conditions initiales. Cependant, nous avons également vérifié la robustesse du dilemme itéré du prisonnier.

Le simulateur logiciel mis au point pour les calculs effectués lors de la thèse a également permis d'étudier deux stratégies nouvelles, aux performances nettement meilleures que celles de `tit_for_tat`. L'étude de ces stratégies a conduit à l'extraction d'un trait important : l'adaptabilité au comportement de l'adversaire. Dans les deux exemples, ces niveaux d'adaptations sont de complexités différentes. Cela nous a donc permis, tout d'abord de confirmer la faiblesse de `tit_for_tat`, de manière plus forte que dans [NS93], mais également d'apporter un nouvel argument contre la simplicité comme qualité des comportements coopératifs.

Afin de conforter cet argument nous avons créé de manière systématique un grand nombre de stratégies. Elles ont été utilisées pour effectuer de nombreuses simulations d'évolutions, impliquant des environnements forts complexes, et hétérogènes : les classes complètes. Grâce à celles-ci nous avons vérifié la force des stratégies adaptatives, et la faiblesse de `tit_for_tat`. Nous avons alors avancé l'idée d'une hiérarchie de stratégies de plus en plus complexe, correspondant à une efficacité croissante. Nous présentons alors la complexité comme un facteur de qualité dans la définition des stratégies pour la coopération entre agents.

Enfin nous avons proposé l'étude d'une variante du dilemme du prisonnier, dans laquelle une certaine forme de communication entre les agents est possible, ou en tout cas envisageable, via l'historique des coups d'une partie. Ce jeu, que nous nommons dilemme de l'ascenceur et qui, dans sa version constitutive, est un dilemme du prisonnier non-pur, se montre intéressant dans le sens où les comportements efficaces qu'il nécessite sont complexes par définition. Il permet également de

modéliser deux niveaux différents de coopération.

Les résultats obtenus tout au long du travail ont été mis à la disposition de la communauté très rapidement via la création d'un site web. Nous pensons en effet que la large diffusion des travaux est un point capital pour la progression de nos connaissances sur les problèmes de coopération entre agents. Les moyens informatiques et de télécommunication actuels permettent de diffuser rapidement et largement la moindre information. Parallèlement au travail de thèse nous avons donc construit un site dans le but de collecter et de rendre disponible le plus d'informations possibles en ce qui concerne la coopération. Cette méthode de diffusion est de plus en plus répandue, mais aussi de plus en plus pertinente, comme le prouve, notamment, la présence sur le réseau de l'encyclopédie de philosophie de Stanford [33].

Aujourd'hui notre site, accessible à l'adresse `http://www.lifl.fr/IPD`, commence à être reconnu par la communauté. Il nous permet de récolter des informations sur la coopération, mais nous oblige également à une grande rigueur dans l'établissement de nos résultats. Cette rigueur a toujours été une volonté forte de Philippe Mathieu et Jean-Paul Delahaye et elle ne nous pose donc pas de problème majeur. La diffusion de nos simulateurs au travers de ce nouveau medium a permis non seulement d'améliorer leurs capacités, mais également de confirmer nos résultats. La maintenance et l'évolution de ce site font donc partie des suites logiques de la thèse.

Cette évolution ne peut-être envisagée que par la confrontation de nos résultats non seulement aux observations biologiques, mais aussi aux applications de ceux-ci dans des cadres plus concrets. Ce dernier point est envisageable dans le cadre de l'utilisation de codage de comportements dans des systèmes multi-agents. L'étude de l'adaptation des stratégies du dilemme du prisonnier à des agents cognitifs est, dans cet esprit, une des suites logiques à ce travail. L'étude du dilemme de l'ascenceur permet également de mieux comprendre le rôle et l'importance de la communication dans des ensembles de tels agents.

D'autre part la poursuite de l'étude de simulations de tailles plus importantes que celles effectuées lors de la thèse est un travail qui a déjà commencé. Le but affiché de ce travail est non seulement de valider encore plus franchement la faiblesse de `tit_for_tat`, et la force de stratégies complexes, mais aussi de tenter de découvrir de nouveaux traits de comportements efficaces.

Finalement la construction de nouvelles stratégies d'adaptation comportementales nous semble être un travail à ne pas négliger. L'amélioration des connaissances théoriques des méthodes d'apprentissages, ou d'adaptation évolutive [34] laisse envisager certaines opportunités de meilleure compréhension des mécanismes de la coopération.

33. `http://plato.stanford.edu`

34. voir par exemple le numéro 229 de Theoretical Computer Science dédié au calcul évolutionnaire [ER99]

Annexe A

Code de stratégies

A.1 Les stratégies classiques

`TURN` correspond au numéro du coup en cours.

`all_c`

Je coopère toujours, quel que soit le comportement de mon adversaire.

`all_d`

Je trahis toujours, quel que soit le comportement de mon adversaire.

`spiteful`

Je coopère jusqu'à ce que mon adversaire ait trahi, après quoi je trahis toujours quel que soit son comportement suivant.

`easy_go`

Je trahis jusqu'à ce que mon adversaire ait coopéré, après quoi je coopère toujours quel que soit son comportement suivant.

`tit_for_tat`

Je coopère au premier coup, ensuite à chaque coup je joue la carte jouée par mon adversaire au coup précédent.

`mistrust`

Je trahis au premier coup, ensuite à chaque coup je joue la carte jouée par mon adversaire au coup précédent.

`two_tit_for_tat`

Je coopère mais trahis deux fois de suite après chaque trahison de mon adversaire.

`three_tit_for_tat`

Je coopère mais trahis trois fois de suite après chaque trahison de mon adversaire.

`tf2t`

Je coopère sauf si mon adversaire a trahi deux fois consécutivement.

`hard_tf2t`

Je coopère sauf si mon adversaire a trahi deux fois consécutivement dans les trois derniers coups.

`slow_tft`

Je coopère deux fois, puis je coopère toujours sauf si mon adversaire a trahi dans l'un des deux derniers coups.

`hard_tft`

Je coopère, puis je coopère toujours sauf si mon adversaire a trahi au moins une fois dans les deux derniers coups.

`soft_majo`

Je joue la carte que mon adversaire a majoritairement jouée dans l'histoire passé de la partie. S'il a joué autant de fois `C`, que `D`, alors je coopère.

`hard_majo`

Je joue la carte que mon adversaire a majoritairement jouée dans l'histoire passé de la partie. S'il a joué autant de fois `C`, que `D`, alors je trahis.

`soft_not_more`

Je trahis si le score de mon adversaire est meilleur que le mien, s'il est égal je coopère.

`soft_not_more_5`

Je trahis si le score de mon adversaire dans les 5 derniers coups est meilleur que le mien, s'il est égal je coopère.

`hard_not_more`

Je trahis si le score de mon adversaire est meilleur ou égal au mien, sinon je coopère.

`hard_not_more_5`

Je trahis si le score de mon adversaire dans les 5 derniers coups est meilleur ou égal au mien, sinon je coopère.

`pavlov`

Je coopère au premier coup puis je coopère uniquement si mon adversaire et moi avons joué la même carte au coup précédent, sinon je trahis.

`prober`

Je commence par jouer la séquence (`DCC`), puis si mon adversaire a coopéré aux coups 2 et 3 alors je joue comme `all_d`, sinon comme `tit_for_tat`.

`ipd_random`

Je coopère et trahis avec une probabilité de 0,5 à chaque coup.

`hard_joss`

Je coopère au premier coup, ensuite si l'adversaire a trahi au coup précédent alors je trahis sinon je trahis avec une probabilité de 10%, et coopère avec une probabilité de 90%.

`soft_joss`

Je coopère au premier coup, ensuite si l'adversaire a trahi au coup précédent alors je trahis avec une probabilité de 90%, et coopère avec une probabilité de 10%, sinon je coopère.

`worse_and_worse`

Je trahis avec une probabilité égale à `TURN`/1000, c'est à dire de plus en plus.

`gradual`

Je coopère jusqu'à la première trahison de mon adversaire. Ensuite pour chaque trahison je punis mon adversaire par n trahison(s), le calme par 2 coopération et continue à coopérer. n est le nombre de fois que mon adversaire m'a trahi.

`gradual_killer`

Je commence par trahir dans les cinq premiers coups, puis je coopère deux fois. Ensuite je trahis tout le temps si mon adversaire a trahi au coup 6 et 7, et coopère tout le temps sinon.

Cette stratégie a été mise au point un vendredi soir tard pour s'assurer que `tit_for_tat` peut battre `gradual` tant en tournoi qu'en évolution. Cela fonctionne dans un environnement de trois stratégies : `tit_for_tat`, `gradual` et `gradual_killer`.

`bad_bet`

Je coopère jusqu'à ce que mon adversaire me trahisse. Ensuite je joue successivement les stratégies `tit_for_tat`, `all_c`, `spiteful`, et `per_ccd`, pendant 4 coups chacune. Puis je joue pendant les 4 coups suivants la stratégie qui m'a rapporté le plus gros score pendant les 4 coups où je l'ai utilisée. Et je recommence ce choix tous les 4 coups.

`c_then_per_dc`

Je coopère jusqu'à la première trahison de mon adversaire après quoi je joue (`DC`)$*$.

`d_then_per_cd`

Je trahis jusqu'à la première trahison de mon adversaire après quoi je joue (`CD`)$*$.

`c_then_per_ccd`

Je coopère jusqu'à la première trahison de mon adversaire après quoi je joue (`CCD`)$*$.

`prober_2`

Je joue (`DCC`) puis si mon adversaire a joué (`DC`) aux coups 2 et 3 alors je coopère tout le temps sinon je joue comme la stratégie `tit_for_tat`.

`prober_3`

Je joue (`DC`) puis si mon adversaire a joué `C` au coup 2 alors je trahis tout le temps sinon je joue comme la stratégie `tit_for_tat`.

`prober_4`

Je joue (`CCDCDDDCCDCDCCDCDDCD`) puis si la différence entre le nombre de trahison de mon adversaire en réponse à une trahison et le nombre de trahison en réponse à une coopération est plus petit que 2 alors je trahis toujours sinon je coopère jusqu'au coup 25 puis je joue comme la stratégie `tit_for_tat`.

`hard_prober`

Je joue (`DDCC`) puis si mon adversaire a coopéré aux coups 2 et 3 alors je trahis, sinon je joue comme la stratégie `tit_for_tat`.

`doubler`

Je coopère toujours sauf si mon adversaire a trahi et que le nombre de coopération de mon adversaire est inférieur à deux fois celui des trahisons.

`soft_spiteful`

Je coopère sauf si mon adversaire viens de trahir, auquel cas je le punie en jouant (`DDDDCC`).

`calculator`

Je joue comme la stratégie `hard_joss` dans les 20 premiers coups. Ensuite si mon adversaire joue périodiquement alors je trahis tout le temps sinon je joue comme la stratégie `tit_for_tat`.

`better_and_better`

Je trahis avec une probabilité de (1000-`TURN`)/1000, c'est-à-dire de moins en moins.

`worse_and_worse`

Je trahis avec une probabilité de `TURN`/1000, c'est-à-dire de plus en plus.

`worse_and_worse2`

Je joue comme la stratégie `tit_for_tat` pendant les 20 premiers coups, puis je trahis avec une probabilité de (`TURN`-20)/`TURN`.

`worse_and_worse3`

Je coopères puis plus mon adversaire trahi, plus je trahis. Au coup `TURN`+1 je trahis avec une probabilité nombre(`D`)/`TURN`.

`reason`

- je joue `C` avec une probabilité de 0,56696 et `D` avec une probabilité de 0,43304 au premier coup et tant que le dernier coup est en phase ;
- ensuite
  - si le premier coup déphasé a été [`C`,`D`], je joue (`DC`)$*$
  - si le premier coup déphasé a été [`D`,`C`], je joue (`CD`)$*$

`naive-reason`

- je joue `C` avec une probabilité 0,56696 et `D` avec une probabilité de 0,43304 au premier coup et tant que le dernier coup est en phase ;
- ensuite
  - si le premier coup déphasé a été [`C`,`D`], je joue (`C`)$*$
  - si le premier coup déphasé a été [`D`,`C`], je joue (`D`)$*$

`reason-tft`

- je joue `C` avec une probabilité 0,56696 et `D` avec une probabilité 0,43304) au premier coup et jusqu'à ce que le dernier coup soit déphasé ;
- ensuite je joue la stratégie `tit_for_tat`.

A.2 Les stratégies génériques

Les stratégies décrites dans le chapitre 5 sont évidemment disponibles en tant que stratégies génériques dans le simulateur. Elles ne sont pas reprises ici.

`per_X`

Je joue périodiquement les coups de la suite X.

`reason_X`

(X est une paramètre entre 0 et 999)
Je joue comme `reason` à l'exception du fait que durant la période de recherche de déphasage, je joue la carte `C` avec la probabilité X/999 et `D` avec la probabilité (999-X)/999.

`reason-tft_X`

(X est une paramètre entre 0 et 999)

- je joue `C` avec une probabilité X/999 et `D` avec une probabilité (999-X)/999 au premier coup et jusqu'à ce que le dernier coup soit en opposition de phase ;
- ensuite je joue la stratégie `tit_for_tat`.

`naive-reason_X`

(X est une paramètre entre 0 et 999)

- je joue `C` avec une probabilité X/999 et `D` avec une probabilité de (999-X)/999 au premier coup et tant que le dernier coup est en phase ;
- ensuite
  - si le premier coup déphasé a été [`C`,`D`], je joue (`C`)$*$
  - si le premier coup déphasé a été [`D`,`C`], je joue (`D`)$*$

Annexe B

Algorithmes de simulations

On trouvera dans cette annexe les versions **simplifiées** des fonctions C utilisées dans le simulateur pour les calculs de tournoi et d'évolutions.

B.1 Tournois

`M(i,j)` correspond au score de la stratégie i face à la stratégie j.

```
di_compute_matrixes ()
{
  /* Matrix scores computations without the diagonal cases */
  di_create_game ();
  for (i = 0; i < size; i += 1)
    {
      for (j = 0; j < i; j += 1)
        {
          game.player_A = di_get_strategy_by_index (list, i);
          game.player_B = di_get_strategy_by_index (list, j);
          play_game (i, j);
        }
    }
  di_destroy_game ();

  /* Diagonal scores computations */
  di_create_game ();
  for (i = 0; i < size; i += 1)
    {
       game.player_A = di_get_strategy_by_index (list, i);
       game.player_B = di_clone_strategy (game.player_A);
       di_play_iterated_game ();
       di_destroy_cloned_strategy (game.player_B);
       M(i,i) = game.score_A;
    }
  di_destroy_game ();
}

di_compute_tournament ()
{
  /* Compute the tournament scores. */
  for (i = 0; i < last; i += 1)
    {
      scores[i] = 0;
      for (j = 0; j < last; j += 1)
        {
          scores[i] += M(i, j);
        }
    }
}
```

B.2 Évolution

`E(i,j)` correspond à la taille de la population de la stratégie i lors de la génération j.

```
void
di_compute_evolution ()
{
  /* Computation of the total size of the population. */
  population_size = 0;
  for (i = 0; i < last; i += 1)
    {
      population_size += E(i,0);
    }

  /* Evolution computation. */
  for (generation = 1;
       is_stabilised (generation - 1) == DI_FALSE && generation < maximum;
       generation += 1)
    {
      /* Strategies's score computation. */
      for (i = 0; i < last; i += 1)
        {
          scores[i] = 0;
          for (j = 0; j < last; j += 1)
            {
              if (i != j)
                {
                  scores[i] += E(j, generation - 1) * M(i, j);
                }
              else
                {
                  scores[i] += (E(j, generation - 1) - 1) * M(i,j);
                }
            }
        }
      /* Total score computation. */
      total_of_scores = 0;
      for (i = 0; i < last; i += 1)
        {
          total_of_scores += scores[i] * E(i, generation - 1);
        }
      /* Proportionnal rate computation. */
      proportionnal_rate = population_size / total_of_scores;
      /* Population redistribution. */
      for (i = 0; i < last; i += 1)
        {
          E(i, generation) = proportionnal_rate * E(i,generation - 1) * scores[i]);
        }
    }
}
```

Annexe C

Résultats des classes complètes

Les résultats présentés ici sont ceux de quelques classes complètes qui ont été simulées. Les résultats comportent pour chaque simulation, les noms des 10 premières stratégies à la fin de l'évolution ainsi que leur population et leur dernière génération d'existence.

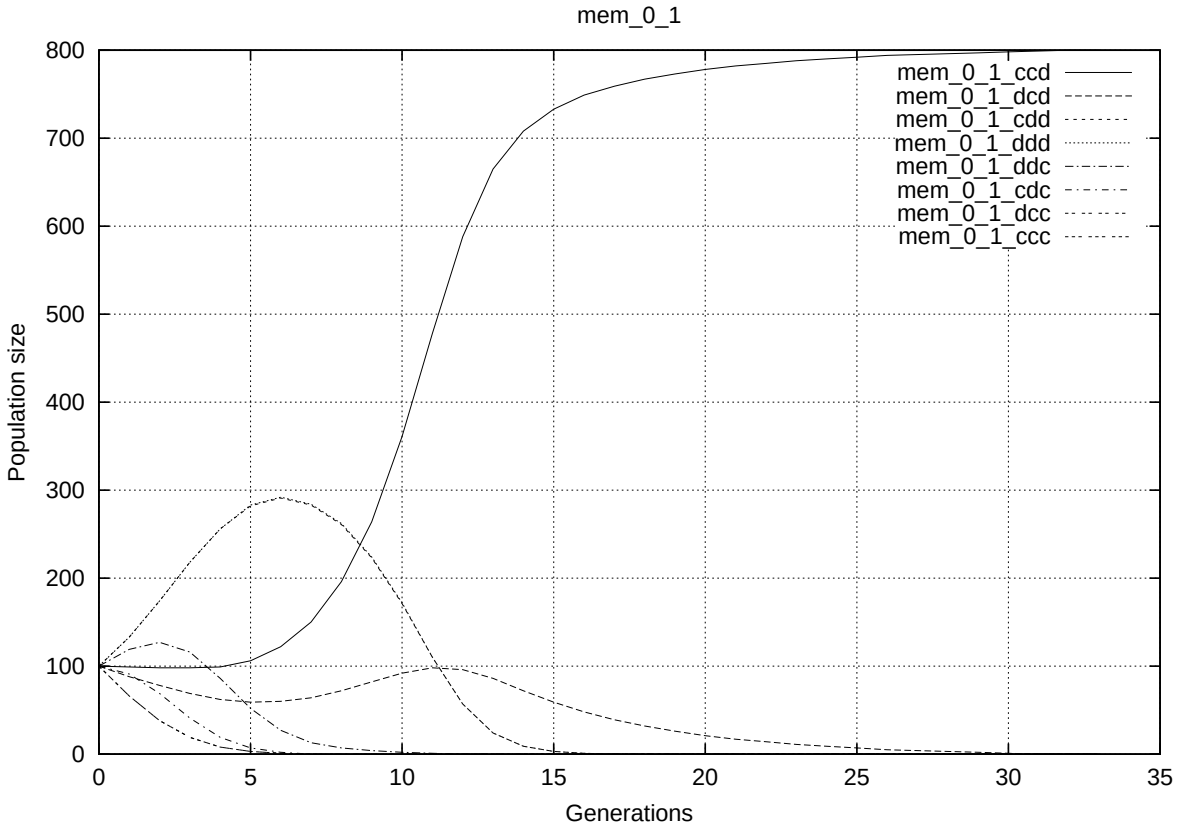
L'évolution des 20 premières stratégies est à chaque fois représentée.

Tous les résultats obtenus ne sont pas exposés ici, mais reste cependant disponibles sur le site du projet PRISON.

C.1 mem_0_1

```
# Needed 0 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       8 strategies:
#           1 :                 mem_0_1_ccd =   800 stopped at generation 33
#           2 :                 mem_0_1_dcd =     0 stopped at generation 30
#           3 :                 mem_0_1_cdd =     0 stopped at generation 16
#           4 :                 mem_0_1_ddd =     0 stopped at generation 16
#           5 :                 mem_0_1_ddc =     0 stopped at generation 11
#           6 :                 mem_0_1_cdc =     0 stopped at generation 6
#           7 :                 mem_0_1_dcc =     0 stopped at generation 6
#           8 :                 mem_0_1_ccc =     0 stopped at generation 6
```

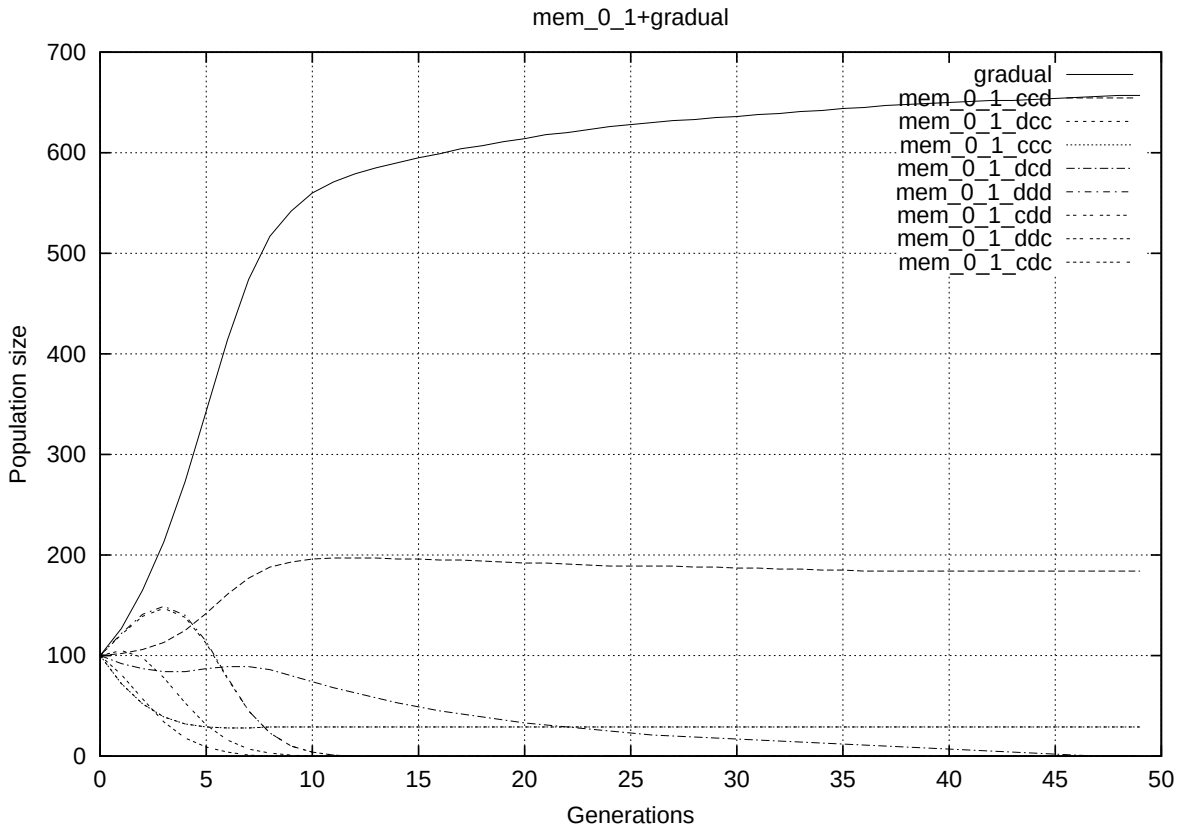
Cette courbe est l'illustration parfaite des évolutions des classes complètes dans la majorité des cas. Au début certaines stratégies profitent de la présence de stratégies facilement exploitables, mais rapidement la disparition de celles-ci profite aux stratégies « *à la* » `tit_for_tat`, qui restent, au début, stable du fait de leur bon comportement face à elles-mêmes.
Sur cet exemple la stratégie gagnante est `tit_for_tat`.



C.2 mem_0_1+gradual

```
# Needed 0 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       9 strategies:
#           1 :                    gradual =   657 stopped at generation 49
#           2 :                mem_0_1_ccd =   184 stopped at generation 49
#           3 :                mem_0_1_dcc =    29 stopped at generation 49
#           4 :                mem_0_1_ccc =    29 stopped at generation 49
#           5 :                mem_0_1_dcd =     0 stopped at generation 46
#           6 :                mem_0_1_ddd =     0 stopped at generation 11
#           7 :                mem_0_1_cdd =     0 stopped at generation 11
#           8 :                mem_0_1_ddc =     0 stopped at generation 9
#           9 :                mem_0_1_cdc =     0 stopped at generation 7
```

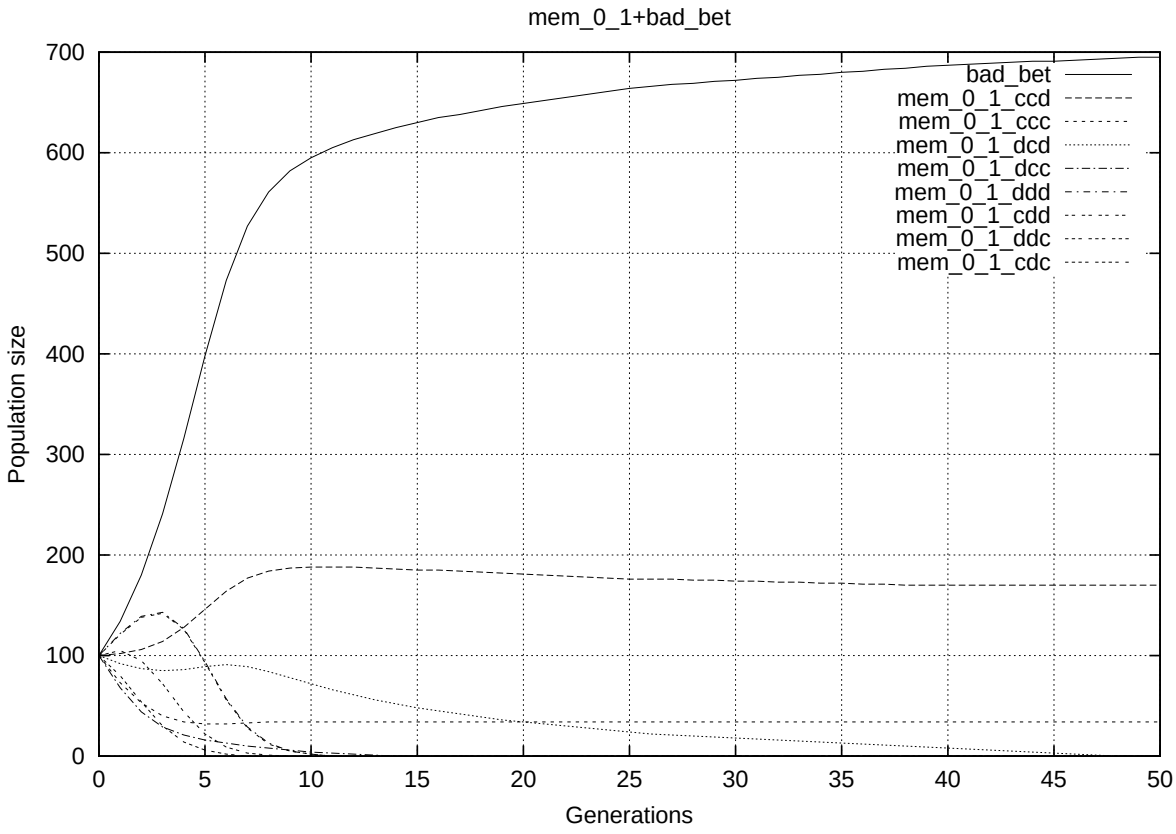
Cette courbe, comme toutes celles de cette classe, est l'illustration parfaite des évolutions des classes complètes impliquant `gradual`. Dès le départ `gradual` prend un avantage indiscutable sur le reste des stratégies. L'évolution se stabilise à peu près de la même manière que dans le cas simple pour les autres stratégies.



C.3 mem_0_1+bad_bet

```
# Needed 0 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       9 strategies:
#           1 :                    bad_bet =   695 stopped at generation 50
#           2 :                mem_0_1_ccd =   170 stopped at generation 50
#           3 :                mem_0_1_ccc =    34 stopped at generation 50
#           4 :                mem_0_1_dcd =     0 stopped at generation 47
#           5 :                mem_0_1_dcc =     0 stopped at generation 13
#           6 :                mem_0_1_ddd =     0 stopped at generation 10
#           7 :                mem_0_1_cdd =     0 stopped at generation 10
#           8 :                mem_0_1_ddc =     0 stopped at generation 8
#           9 :                mem_0_1_cdc =     0 stopped at generation 6
```

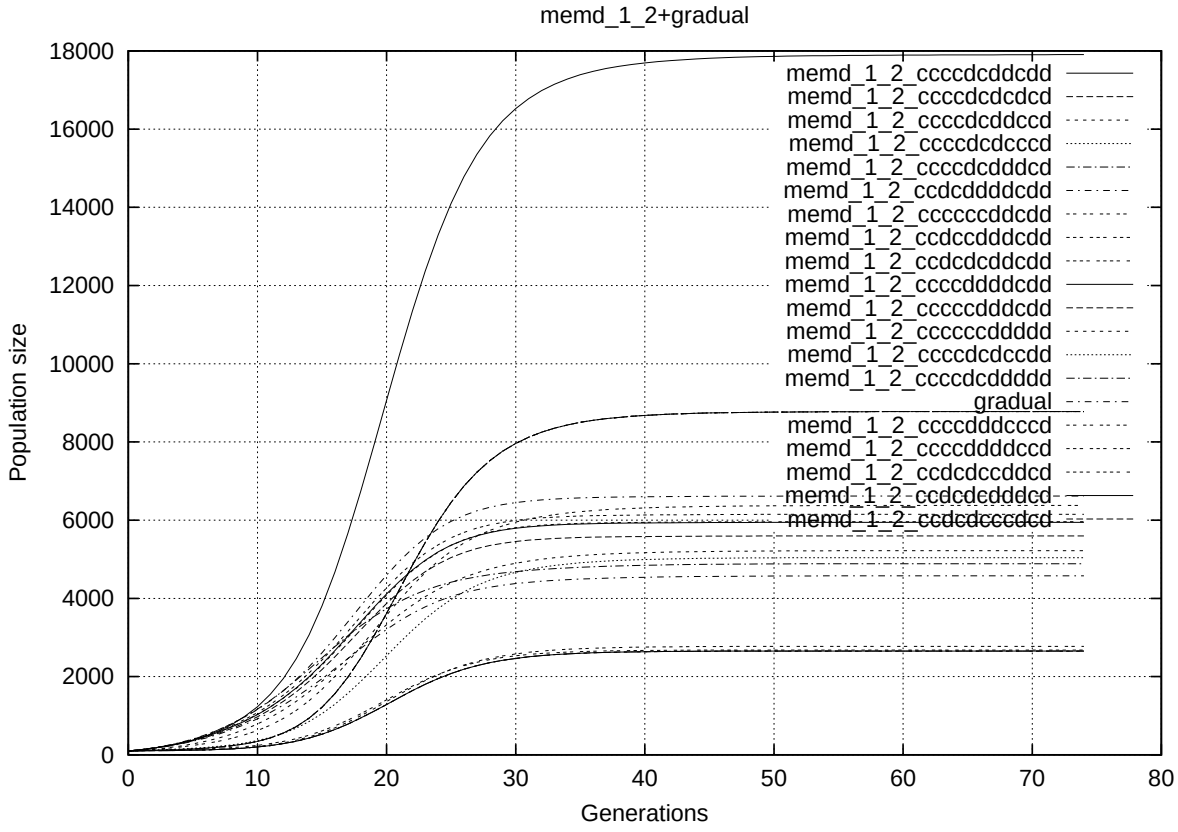
L'évolution est dans ce cas souvent similaire à la précédente impliquant `gradual`, mais avec une pente de progression beaucoup plus importante pour `bad_bet` que pour `gradual`.
De plus l'écart entre le premier et le second est beaucoup plus conséquent.



C.4 `memd_1_2+gradual`

```
# Needed 155 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       2049 strategies:
#           1 :        memd_1_2_ccccdcddcdd = 17905 stopped at generation 74
#           2 :        memd_1_2_ccccdcdcdcd =  8777 stopped at generation 74
#           3 :        memd_1_2_ccccdcddccd =  8777 stopped at generation 74
#           4 :        memd_1_2_ccccdcdcccd =  8777 stopped at generation 74
#           5 :        memd_1_2_ccccdcdddcd =  8777 stopped at generation 74
#           6 :        memd_1_2_ccdcddddcdd =  6617 stopped at generation 74
#           7 :        memd_1_2_ccccccddcdd =  6375 stopped at generation 74
#           8 :        memd_1_2_ccdccdddcdd =  6150 stopped at generation 74
#           9 :        memd_1_2_ccdcdcddcdd =  5961 stopped at generation 74
#          10 :        memd_1_2_ccccddddcdd =  5945 stopped at generation 74
```

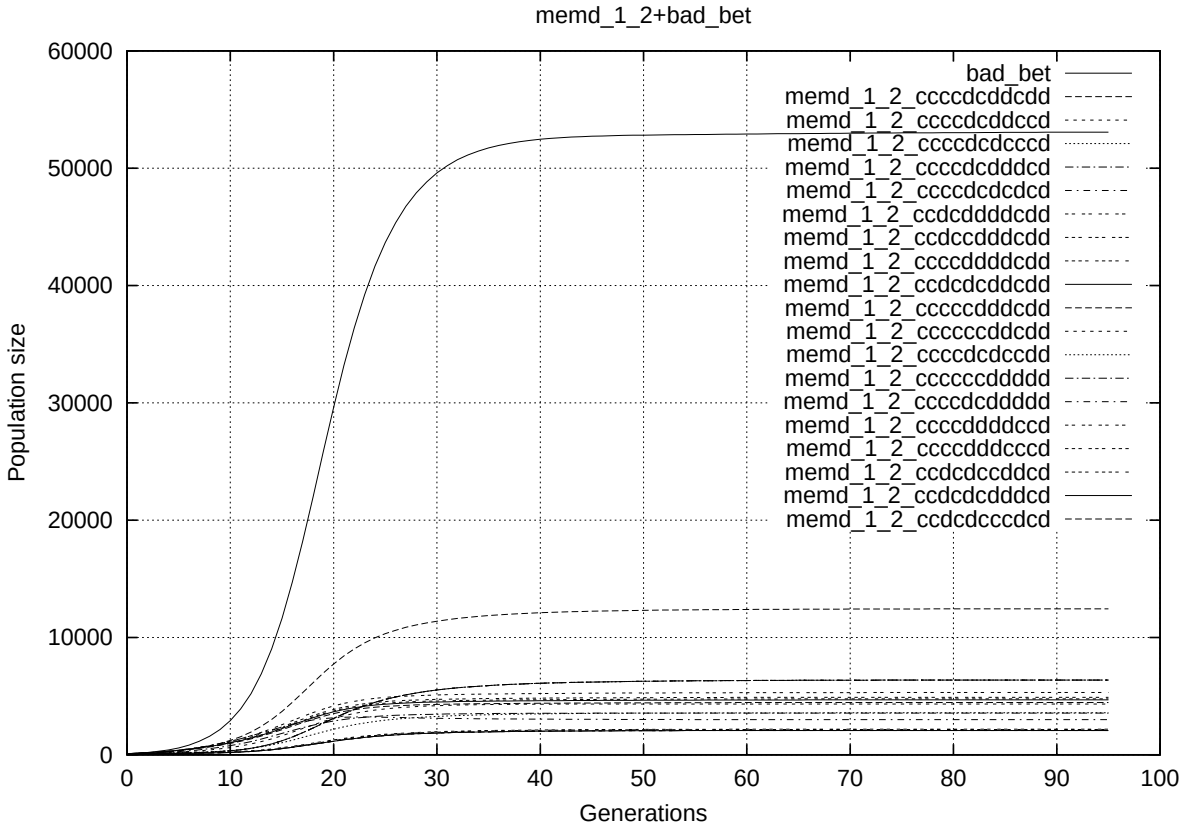
On constate sur cette évolution et la suivante le fait que `gradual` est moins performante que `bad_bet`, mais que malgré cela l'évolution n'amène pas de dynamique vraiment différente des cas précédent.
À chaque fois une stratégie sort très nettement du lot à la fin de l'évolution, en occupant une très grosse partie de la population complète, souvent plus de 20%.



C.5 memd_1_2+bad_bet

```
# Needed 180 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       2049 strategies:
#           1 :                      bad_bet = 53073 stopped at generation 95
#           2 :        memd_1_2_ccccdcddcdd = 12446 stopped at generation 95
#           3 :        memd_1_2_ccccdcddccd =  6371 stopped at generation 95
#           4 :        memd_1_2_ccccdcdcccd =  6371 stopped at generation 95
#           5 :        memd_1_2_ccccdcdddcd =  6371 stopped at generation 95
#           6 :        memd_1_2_ccccdcdcdcd =  6371 stopped at generation 95
#           7 :        memd_1_2_ccdcddddcdd =  5308 stopped at generation 95
#           8 :        memd_1_2_ccdccdddcdd =  4906 stopped at generation 95
#           9 :        memd_1_2_cccddddcdd =  4770 stopped at generation 95
#          10 :        memd_1_2_ccdcdcddcdd =  4682 stopped at generation 95
```

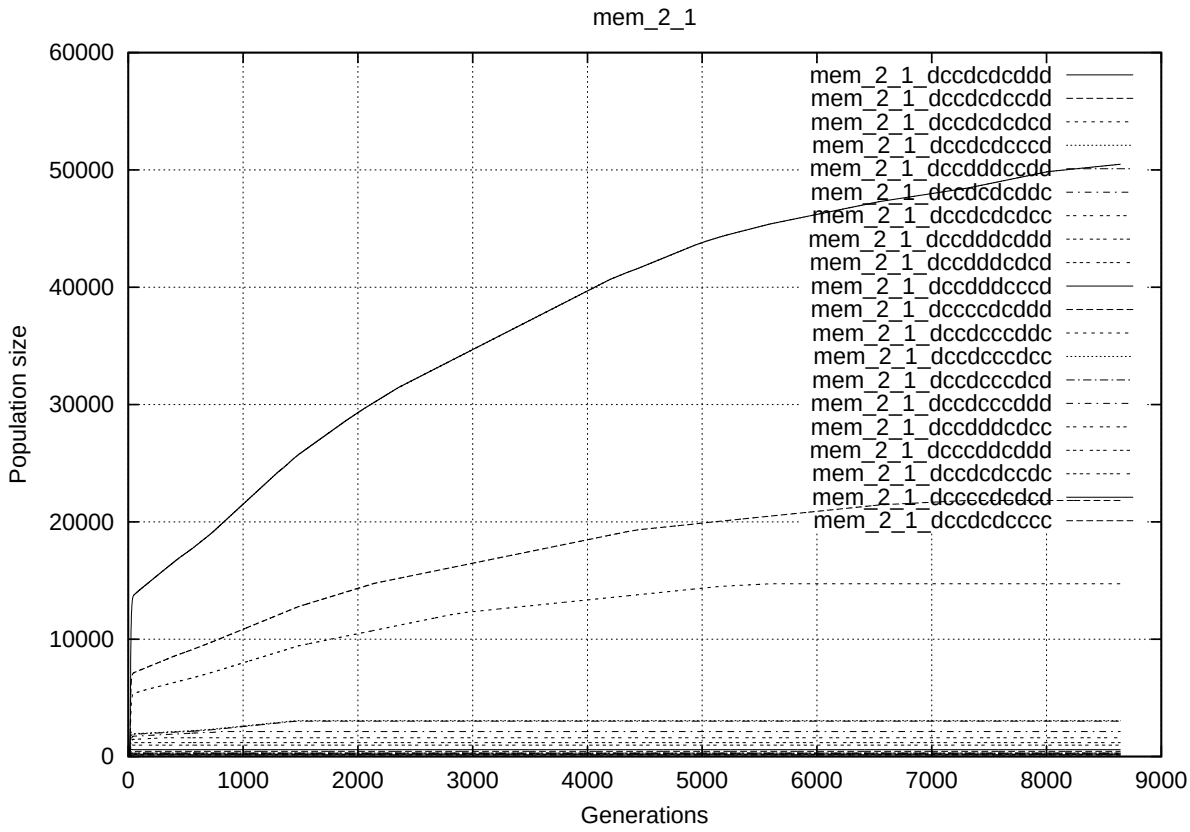
Il est aussi intéressant de noter que l'ajout de la dynamicité n'apporte pas de grands changements dans les qualités des stratégies, si ce n'est que les évolutions sont un peu plus longues. Les premières générations sont plus stables que dans le cas général. La dynamique prend sa forme définitive beaucoup plus tard.
Ici, encore M < O et de ce fait le début de partie n'a pas une grande incidence sur le comportement.



C.6 mem_2_1

```
# Needed 3385 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       1024 strategies:
#           1 :          mem_2_1_dccdcdcddd = 50494 stopped at generation 8647
#           2 :          mem_2_1_dccdcdccdd = 21835 stopped at generation 8647
#           3 :          mem_2_1_dccdcdcdcd = 14724 stopped at generation 8647
#           4 :          mem_2_1_dccdcdcccd =  3059 stopped at generation 8647
#           5 :          mem_2_1_dccdddccdd =  3015 stopped at generation 8647
#           6 :          mem_2_1_dccdcdcddc =  2133 stopped at generation 8647
#           7 :          mem_2_1_dccdcdcdcc =  1599 stopped at generation 8647
#           8 :          mem_2_1_dccdddcddd =  1192 stopped at generation 8647
#           9 :          mem_2_1_dccdddcdcd =   977 stopped at generation 8647
#          10 :          mem_2_1_dccdddcccd =   588 stopped at generation 8647
```

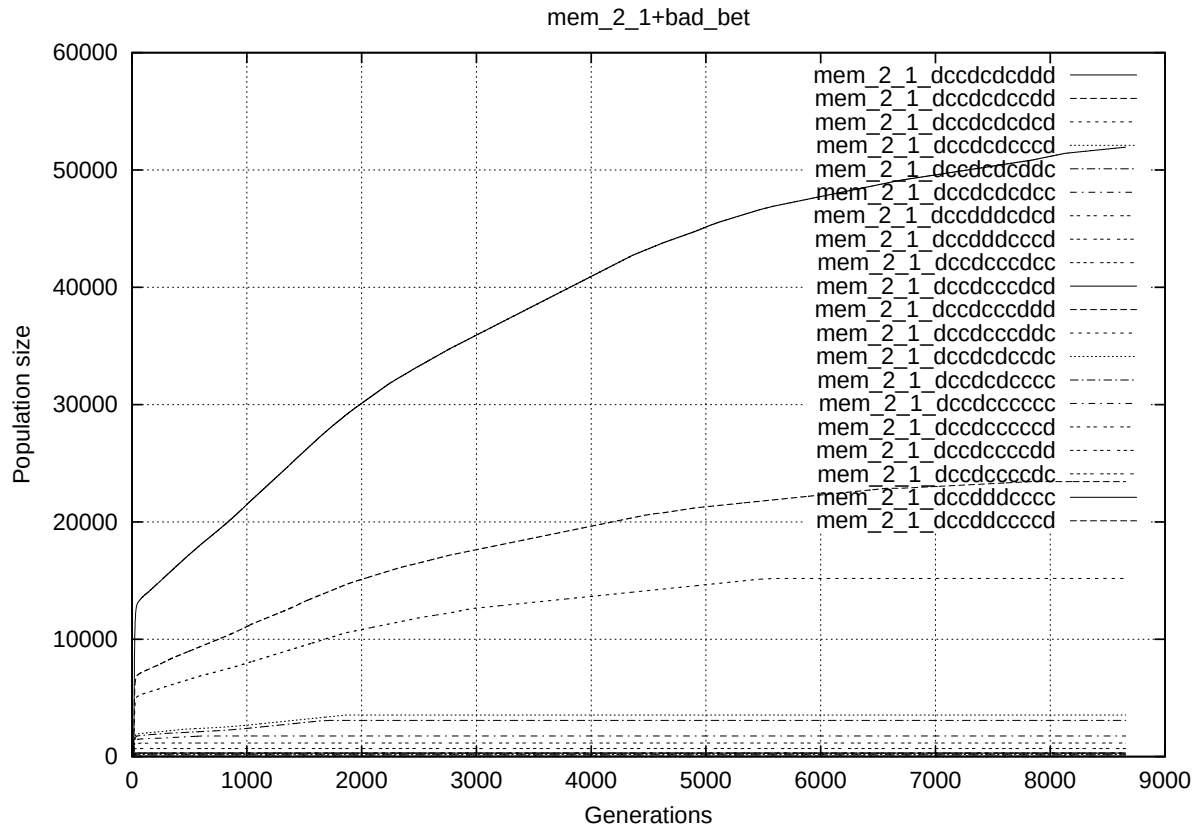
On voit ici un exemple d'évolution d'une classe complète pour laquelle M > O. Dans ce genre d'évolution les stratégies `bad_bet`, comme `gradual`, n'arrivent pas à bien évoluer. Le fait que les stratégies de ces classes prennent plus en compte les coups qu'elles jouent plutôt que les coups que leurs adversaires jouent est un grave handicap pour les stratégies réactives comme `bad_bet`. En effet le début d'une partie prend ici une importance capitale pour, par exemple, `bad_bet`.
Il faut aussi noter que les évolutions sont de plus en plus longues, et que les calculs demandent, de ce fait beaucoup plus de temps.



mem_2_1+bad_bet

```
# Needed 3373 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       1025 strategies:
#            1 :          mem_2_1_dccdcdcddd = 51958 stopped at generation 8661
#            2 :          mem_2_1_dccdcdccdd = 23446 stopped at generation 8661
#            3 :          mem_2_1_dccdcdcdcd = 15182 stopped at generation 8661
#            4 :          mem_2_1_dccdcdcccd =  3539 stopped at generation 8661
#            5 :          mem_2_1_dccdcdcddc =  3090 stopped at generation 8661
#            6 :          mem_2_1_dccdcdcdcc =  1751 stopped at generation 8661
#            7 :          mem_2_1_dccdddcdcd =  1153 stopped at generation 8661
#            8 :          mem_2_1_dccdddcccd =   679 stopped at generation 8661
#            9 :          mem_2_1_dccdcccdcc =   298 stopped at generation 8661
#           10 :          mem_2_1_dccdcccdcd =   298 stopped at generation 8661
```

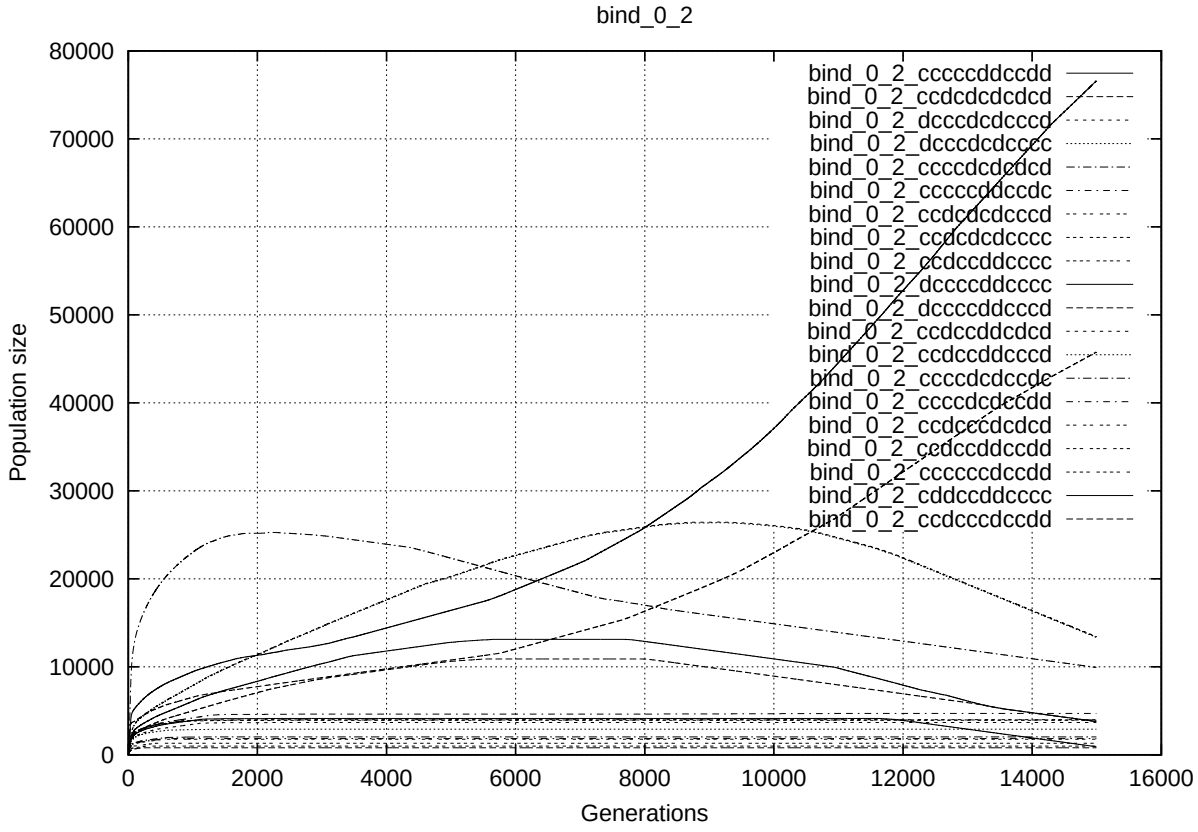
L'introduction de `bad_bet` ne modifie que très peu l'évolution de la classe complète.



C.7 `bind_0_2`

```
# Needed 12575 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       2048 strategies:
#           1 :        bind_0_2_cccccddccdd = 76626 stopped at generation 14999
#           2 :        bind_0_2_ccdcdcdcdcd = 45787 stopped at generation 14999
#           3 :        bind_0_2_dcccdcdcccd = 13419 stopped at generation 14999
#           4 :        bind_0_2_dcccdcdcccc = 13358 stopped at generation 14999
#           5 :        bind_0_2_ccccdcdcdcd =  9931 stopped at generation 14999
#           6 :        bind_0_2_cccccddccdc =  4710 stopped at generation 14999
#           7 :        bind_0_2_ccdcdcdcccd =  3997 stopped at generation 14999
#           8 :        bind_0_2_ccdcdcdcccc =  3960 stopped at generation 14999
#           9 :        bind_0_2_ccdccddcccc =  3916 stopped at generation 14999
#          10 :        bind_0_2_dccccddcccc =  3784 stopped at generation 14999
```

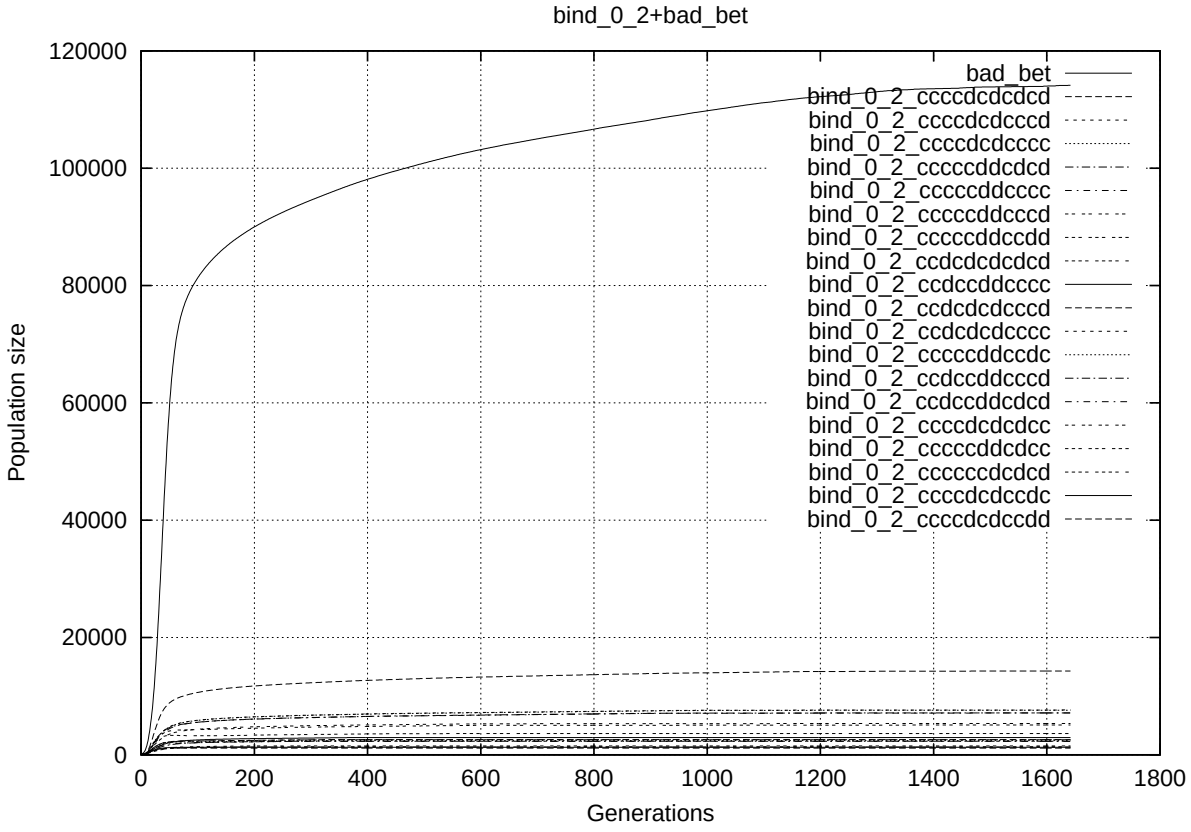
Cette évolution n'est pas terminée. Nos simulations n'autorisaient, pour des raisons d'espace mémoire disponibles, aucune évolution supérieure à 15 000 générations.
On imagine cependant très bien le scénario de cette évolution. Il ressemble au premier cas décrit avec un simple changement d'échelle.



C.8 bind_0_2+bad_bet

```
# Needed 2671 seconds to compute the evolution.
# ECOLOGICAL EVOLUTION RANK
#       2049 strategies:
#           1 :                      bad_bet = 114131 stopped at generation 1642
#           2 :        bind_0_2_ccccdcdcdcd = 14286 stopped at generation 1642
#           3 :        bind_0_2_ccccdcdcccd =  7643 stopped at generation 1642
#           4 :        bind_0_2_ccccdcdcccc =  7611 stopped at generation 1642
#           5 :        bind_0_2_cccccddcdcd =  7152 stopped at generation 1642
#           6 :        bind_0_2_cccccddcccc =  7133 stopped at generation 1642
#           7 :        bind_0_2_cccccddcccd =  5354 stopped at generation 1642
#           8 :        bind_0_2_cccccddccdd =  5079 stopped at generation 1642
#           9 :        bind_0_2_ccdcdcdcdcd =  3621 stopped at generation 1642
#          10 :        bind_0_2_ccdccddcccc =  2954 stopped at generation 1642
```

Pour finir il est très intéressant de noter la force de `bad_bet` qui accélère l'évolution de manière phénoménale, par rapport au résultat précédent de la même classe complète.
Il faut noter qu'en dix fois moins de temps que pour l'évolution de la classe complète `bad_bet` a envahi un peu plus de la moitié de la population totale.



Bibliographie

[AD88] Robert Axelrod and Douglas Dion. The further evolution of cooperation. *Science*, 242:1385–1390, december 1988.

[AH81] Robert Axelrod and Williamm D. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, march 1981.

[AH92] Robert J. Aumann and Sergiu Hart, editors. *Handbook of game theory with economic applications*, volume 1. North-Holland, 1992.

[AH94] Robert J. Aumann and Sergiu Hart. *Handbook of game theory with economic applications*, volume 2. North-Holland, 1994.

[Axe84] Robert Axelrod. *The evolution of cooperation.* Basic Books, New-York, USA, 1984.

[Axe92] Robert Axelrod. *Donnant donnant : théorie du comportement coopératif.* Éditions Odile Jacob, Paris, France, 1992. Traduction française de [Axe84], Réédité en [Axe96].

[Axe96] Robert Axelrod. *Comment réussir dans un monde d'égoïstes.* Éditions Odile Jacob, Paris, France, 1996.

[Axe97] Robert Axelrod. *The Complexity of Cooperation.* Princeton University Press, Priceton, NJ, USA, 1997.

[BDM96] Bruno Beaufils, Jean-Paul Delahaye, and Philippe Mathieu. Our meeting with gradual, a good strategy for the iterated prisoner's dilemma. In Christopher G. Langton and Katsunori Shimohara, editors, *Proceedings of Artificial Life V*, pages 202–209, Cambridge, MA, USA, 1996. The MIT Press/Bradford Books. Artificial Life V, Nara, Japan, May 16-18 199.

[BDM98] Bruno Beaufils, Jean-Paul Delahaye, and Philippe Mathieu. Complete classes of strategies for the classical iterated prisoner's dilemma. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionnary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 33–41. Springer-Verlag, 1998. Evolutionnary Programing VII, San Diego, CA, USA, March 25-27, 1998.

[Bin99] Ken Binmore. *Jeux et théorie des jeux.* DeBoeck, 1999.

[BL87] Robert Boyd and Jeffrey P. Loberbaum. No pure strategy is evolutionarily stable in the repeated prisoner's dilemma game. *Nature*, 327:58–59, 1987.

[BS95] Jonathan Bendor and Piotr Swistak. Types of evolutionary stability and the problem of cooperations. *Proceedings of the National Academy of Sciences*, 92:3596–3600, april 1995.

[BS96a] Jonathan Bendor and Piotr Swistak. The controversy about the evolution of cooperatoin and the evolutionary roots of social institutions. In *Social Agency.* Transaction Publishers, New Brunswick, NJ, USA, 1996.

[BS96b] Jonathan Bendor and Piotr Swistak. Evolutionary equilibria: Characterization theorems and their implications. *Theory and Decision*, 3, 1996.

[BS97] Jonathan Bendor and Piotr Swistak. The evolutionary stability of cooperations. *American Political Science Review*, 91(2):290–307, june 1997.

[Cor97] Laurent Cordonnier. *Coopération et Réciprocité.* Sociologies. Presses Universitaires de France, 1 edition, mars 1997.

[Dav87] L. Davis, editor. *Genetic Algorithms and the Simulated Annealing*, chapter 3: The Evolution of Strategies in the Iterated Prisoner's Dilemma. Pitman, London, 1987. The author is Robert Axelrod.

[Del97] Samuel Delepoulle. La coopération entre agents adaptatifs : Préambule à une approche sélectionniste des comportements sociaux. Mémoire de dea, Université de Lille 3 – UFR de Psychologie, juin 1997.

[DM92] Jean-Paul Delahaye and Philippe Mathieu. Expériences sur le dilemme itéré des prisonniers. Publication interne IT-233, Laboratoire d'Informatique Fondamentale de Lille (LIFL), 1992.

[DM93] Jean-Paul Delahaye and Philippe Mathieu. L'altruisme perfectionné. Publication interne IT-249, Laboratoire d'Informatique Fondamentale de Lille (LIFL), 1993.

[DM95] Jean-Paul Delahaye and Philippe Mathieu. Complex strategies in the iterated prisoner's dilemma. In A. Albert, editor, *Chaos and Society*. IOS Press, 1995.

[DM96] Jean-Paul Delahaye and Philippe Mathieu. Random strategies in a two levels iterated prisoner's dilemma: How to avoid conflicts ? In H. Jürgen Müller and Rose Dieng, editors, *Proceedings of the ECAI'96 Workshop (W24) : Modelling Conflicts in AI*, pages 68–72. European Coordinating Comitte for Artificial Intelligence (ECCAI), 1996. 12th European Conference on Artificial Intelligence, 11-16 August 1996, Budapest, Hungary.

[ER99] A.E. Eiben and G. Rudolph. Theory of evolutionary algorithms: a bird's eye view. *Theoretical Computer Science*, 229:2–9, 1999.

[Fer95] Jacques Ferber. *Les systèmes multi-agents : vers une intelligence collective*. InterÉditions, 1995.

[Flo52] Merrill M. Flood. Some experimental games. Research Memorandum RM-789-1, The RAND Corporation, Santa-Monica, CA, USA, 1952.

[Fog93] David B. Fogel. Evolving behaviors in the iterated prisoner's dilemma. *Evolutionary Computation*, 1(1):77–97, 1993.

[Fog95] David B. Fogel. *Evolutionary computation : Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.

[Fog96] David B. Fogel. Special issue on the prisoner's dielmma. *Biosystems*, 37(1,2):1–176, 1996.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[Gue95] Bernard Guerrien. *La Théorie des Jeux*. Economica, 2 edition, 1995.

[How66] N. Howard. The theory of meta-games. *General Systems*, 11(5):167–186, 1966.

[HP95] Robert Heinhsohn and Craig Packer. Complex cooperative strategies in group-territorial african lions. *Science*, 269:1260–1262, september 1995.

[HS88] Josef Hofbauer and Karl Sigmund. *Theory of Evolution and Dynamical Systems*. Cambridge University Press, 1988.

[HS98] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[Jay96] Hubert Jayet. Introduction à la théorie des jeux et à ses applications économiques. Support de cours de théorie des jeux, Licence de Sciences Économiques, Facultés de Sciences Économiques et Sociales, Université des Sciences et Technologies de Lille, 1995–1996. Attention, les versions disponibles en 1999 comportent de nombreuses erreurs.

[JG73] Maynard Smith J. and Price G.R. The logic of animal conflict. *Nature*, 246:15–18, 1973.

[KK89] David Kraines and Vivian Kraines. Pavlov and the prisoner's dilemma. *Theory and Decision*, 26:47–79, 1989.

[Kuh99] Steven T. Kuhn. Prisoner's dilemma. In Edward N. Zalta, editor, *Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab at the Center for the Study of Language and Information, Stanford University, Stanford, CA, USA, fall 1999. C'est un site web : `http://plato.stanford.edu/fall1999/entries/prisoner-dilemma/`.

[Lan89] Christopher G. Langton. Artificial life. In Christopher G. Langton, editor, *Proceedings of Artificial Life*, Santa Fe Institute Studies in the Sciences of Complexity, pages 1–47, Reading, MA, USA, 1989. Addisson–Wesley Publishing Company. Artificial Life, Los Alamos, USA, September 1987.

[Lin92] Kristian Lindgren. Evolutionary phenomea in simple dynamics. In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Proceedings of Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, pages 295–312, Reading, MA, USA, 1992. Addisson–Wesley Publishing Company. Artificial Life II, Santa Fe, USA, February 1990.

[LN95] K. Lindgren and M. G. Nordhal. Cooperation and community structure in artificial ecosystems. In C. G. Langton, editor, *Artificial Life, An Overview*, chapter Cambridge, MA, USA, pages 15–37. The MIT Press, 1995.

[LR85] R. Duncan Luce and Howard Raiffa. *Games Decisions : Introduction and Critical survey*. Dover Publications, 2 edition, 1985. Cette référence correspond à la seconde édition de l'ouvrage, la première datant de 1957.

[MCWG95] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[MD99] H. Jürgen Muller and Rose Dieng, editors. *Computational conflicts*, chapter 11, pages 192–213. Springer-Verlag, 1999.

[MDB99] Philippe Mathieu, Jean-Paul Delahaye, and Bruno Beaufils. Studies on dynamics in the classical iterated prisoner's dilemma with few strategies: Is there any chaos in the pure dilemma? In *Évolution Artificielle 1999*, 1999. Évolution Artificielle 1999, Dunkerque, France, 3-5 Novembre 1999.

[Mil96] John H. Miller. The coevolution of automate in the repeated prisoner's dilemma. *Journal of Economic Behavior and Organization*, 29:87–112, 1996.

[ML96] Nicolas Meuleau and Claude Lattaud. The artificial evolution of cooperation. In Alliot J.-M., Lutton M., Ronald E., Schoenauer M., and Snyers D., editors, *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science*, pages 159–180, Berlin, 1996. Springer-Verlag.

[Mor95] Virginia Morell. Cowardly lions confound cooperation theory. *Science*, 269:1216–1217, september 1995.

[Nac92] John H. Nachbar. Evolution in the finitely repeated prisoner's dilemma. *Journal of Economic Behavior and Organization*, 19:307–326, 1992.

[Now90] Martin Nowak. Stochastic strategies in the prisoner's dilemma. *Theoretical Population Biology*, 38:93–112, 1990.

[NS89] Martin Nowak and Karl Sigmund. Oscillations in the evolution of reciprocity. *Journal of Theroetical Biology*, 137:21–26, 1989.

[NS90] Martin Nowak and Karl Sigmund. The evolution of stochastic strategies in the prisoner's dilemma. *Acta Applicandae Mathematicae*, 20:247–265, 1990.

[NS93] Martin Nowak and Karl Sigmund. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, 364:56–58, july 1993.

[NS99a] Martin A. Nowak and Karl Sigmund. Aux racines de la coopération. *La recherche*, 325:38–39, novembre 1999. Traduction de [NS99b].

[NS99b] Martin A. Nowak and Karl Sigmund. Phage-lift for game theory. *Nature*, 398:367–368, april 1999.

[Poo95] Robert Pool. Putting game theory to the test. *Science*, 267:1591–1593, march 1995.

[Pou92] William Poundstone. *Prisoner's Dilemma*. Doubleday, 1992.

[RC65] Anatol Rapoport and Albert M. Chammah. *Prisoner's Dilemma: A Study in Conflict and Cooperation*. The University of Michigan Press, Ann Arbor, 1965.

[RG66] Anatol Rapoport and Melvin Guyer. A taxonomy of 2x2 games. *General Systems*, 11:203–214, 1966.

[Shu64] Martin Shubik. *Related Approaches to Social Game Theory and Behavior*. Wiley, New-York, USA, 1964.

[Shu70] Martin Shubik. Game theory, behavior, and the paradox of the prisoner's dilemma: three solutions. *Journal of Conflict resolution*, XIV(2):181–193, 1970. Journal of Conflict Resolution web site is at `http://www.library.yale.edu/un/un2f1a1.htm`.

[Smi82] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[Smi96] John Maynard Smith. The games the lizards play. *Nature*, 380:198–199, 1996.

[TC99] Paul E. Turner and Lin Chao. Prisoner's dilemma in rna virus. *Nature*, 398:441–443, april 1999.

[vBvKP99] D.D.B. van Bragt, C.H.M. van Kemenade, and J.A. La Poutré. The influence of evolutionary selection schemes on the iterated prisoner's dilemma. In *Proceedings of Computing in Economics and Finance Conference (CEF99)*, 1999. Computing in Economics and Finance Conference, Boston College, Chestnut Hill, MA, June 24-26, 1999.

[vNM44] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. The standard reference is the revised edition of 1947.

version du lundi 15 avril 2002 à 17 h 27

Résumé

Le dilemme itéré du prisonnier est une représentation mathématique de la coopération entre agents. Ce modèle est issu de la théorie des jeux dont le but initial est d'étudier les situations de conflits d'intérêts entre individus. La pauvreté des résultats qu'elle implique dans le cas du dilemme du prisonnier rend son utilisation assez inefficace. Une nouvelle approche évolutionniste basée en grande partie sur des simulations informatiques a été initiée par Robert Axelrod. Les agents sont caractérisés par leur comportement, ou stratégie. Axelrod a mis en évidence quatre propriétés qu'une stratégie doit posséder pour être efficace, et propose la stratégie `donnant_donnant` comme exemple.

Notre travail consiste à étudier et approfondir ce type de simulations. Nous adaptons le modèle afin de prendre en compte l'aspect discret des calculs. Cette adaptation nous permet de faire un grand nombre de simulations confirmant en majeure partie les résultats obtenus dans le cas continu. Ceci remet cependant en cause une des propriétés avancées par Axelrod : la simplicité. Nous illustrons ceci par la présentation de stratégies meilleures que `donnant_donnant` et à complexité plus importante. Les évaluations sont faites grâce à des simulations impliquant un très grand nombre de stratégies construites de manière objective via une approche génétique.

Ces simulations permettent de mettre en évidence une nouvelle propriété : la faculté d'adaptation du comportement. Cette nouvelle propriété renforce l'idée de complexité croissante dans les comportements coopératifs entre agents.

Nous débutons également l'étude d'un dilemme du prisonnier particulier dont seule l'itération diffère du modèle classique et qui permet de modéliser deux niveaux de coopération : le dilemme de l'ascenseur. Cette étude théorique et expérimentale nous permet de montrer qu'avec cette nouvelle représentation les comportements purement déterministes ne peuvent être efficaces.

Mots-clés: Dilemme (itéré) du prisonnier, Coopération, Évolution de la coopération, Théorie des jeux, Vie artificielle, Intelligence artificielle, Systèmes multi-agents, Jeu répété

Abstract

The iterated prisoner's dilemma is a cooperation between agents mathematical representation. This model comes from game theory, which initial goal is to study conflict of interest situations. Unfortunately results implied by the theory on the prisoner's dilemma are so poor that it could not be used seriously. A new evolutionary approach, mainly based on computer simulations, has been initiated by Robert Axelrod. Agents are characterized by their behavior, or strategy. Axelrod brings to the fore four properties a strategy has to respect in order to be efficient, and propose the `tit_for_tat` strategy as an example.

Our work is to study and increase knowledge on those kind of simulations. We adapt the model such that it takes into account the fact that computing is a discrete job. This adaptation allows us to make a lot of simulations which confirm main results obtained in the continuous case. This, however, call into question one of Axelrod property : simplicity. We describe strategies more efficient than `tit_for_tat` but with a higher level of complexity. Evaluations are done by simulations involving a big number of strategies build objectively via a genetic approach.

Those simulations allow us to find a new property : behavior adaptation faculty. This new property enforce the idea that cooperative behavior between agents may be more and more complex.

We also start the study of a particular prisoner's dilemma, which differs from the classical model only in the iterated version and which allows to represent two level of cooperation : the lift dilemma. This theoretical as well as experimental study allow us to show that with this new model no purely deterministic behavior can be efficient.

Keywords: (Iterated) Prisoner's dilemma, Cooperation, Evolution of cooperation, Game theory, Artificial life, Artificial intelligence, Multi-agents system, Repeated game