Mémoire de DEA — Session de juillet 2000

Evaluation de comportements adaptatifs entre agents dans des méta-jeux

François Delabre

Sous la direction de
Philippe Mathieu

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
Laboratoire d'Informatique Fondamentale de Lille – UPRESA 8022 CNRS
UFR d'IEEA Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. +33 (0)3 20 43 47 24 – Télécopie +33 (0)3 20 43 65 66 – E-mail direction@lifl.fr

Table des matières

Introduction

Le dilemme du prisonnier, problème bien connu de la théorie des jeux, modélise les interactions qui peuvent avoir lieu entre des personnes. Ces personnes, ou joueurs, peuvent coopérer ou trahir, et obtiennent un score en fonction de leur comportement et de celui de leur adversaire.

Dans ce projet, nous étudions différentes versions étendues du dilemme, que nous appelons méta-jeux. Alors que, dans le dilemme du prisonnier, les scores obtenus restent fixes tout au long d'une partie, ils peuvent, dans le cadre des méta-jeux, varier aléatoirement. De plus, les comportements ne se limitent pas à coopérer ou trahir : les joueurs ont un nombre variable de choix de comportements. Ainsi, les méta-jeux tentent de modéliser les interactions entre personnes dans un cadre dynamique où les règles peuvent changer.

Différentes stratégies devront être définies pour ces méta-jeux, soit en s'appuyant sur les résultats du dilemme du prisonnier, soit en imaginant de nouvelles tactiques. L'objectif est, à l'aide d'expérimentations, de mettre en évidence la nécessité, pour une bonne stratégie, d'analyser les règles du jeu, de s'y adapter et de jouer le mieux possible contre elle-même. Il serait également intéressant d'essayer de montrer qu'il existe de meilleures stratégies que celles qui jouent les équilibres de Nash, ce qui est considéré comme la façon raisonnable de jouer. Enfin, un système permettant de décrire facilement une stratégie permettrait de créer un site de concours en ligne.

Chapitre 1

Les méta-jeux

1.1 Contexte

Ce travail s'appuie sur celui réalisé par Bruno Beaufils dans [1]. Le cadre est celui de la théorie des jeux, décrite entre autres dans [2], [3]. Plus précisément, il s'intéressait à la coopération entre agents, en particulier au dilemme du prisonnier ainsi qu'au dilemme de l'ascenseur. Pour des précisions sur ces thèmes, voir [4], [5].

Ici, on se propose d'étudier un jeu différent, qui est plus général que celui du dilemme du prisonnier. Le but est à la fois d'essayer de transposer les résultats obtenus sur le dilemme du prisonnier, de tenter de les généraliser, et d'en trouver de nouveaux.

1.2 Règles du jeu

1.2.1 Jeu simple

Le jeu cherche à modéliser les interactions que l'on peut avoir avec les gens que l'on ne rencontre qu'une seule fois. Pour cela, on considère un jeu à deux joueurs qui jouent simultanément un coup parmi n.

Le jeu est symétrique, ce qui signifie que les deux joueurs jouent exactement dans les mêmes conditions. Ils ont les mêmes choix et obtiennent le même nombre de points dans des circonstances identiques.

Le jeu n'est ni à somme nulle, ni à somme constante. La somme des points marqués par les deux joueurs peut varier en fonction des coups qu'ils ont choisis.

Les scores sont déterminés par une matrice de gains de taille $n \times n$. Cette matrice contient les scores qu'un joueur obtient contre son adversaire en fonction des deux coups joués. Elle est tirée aléatoirement uniformément entre 0 et k au début du jeu.

U	A	B	C
A	6	2	3
B	8	5	7
C	9	8	2

FIG. 1.1 – Exemple de matrice de gains U avec n = 3 et k = 9

Cette matrice est connue des deux joueurs. Le gagnant est celui qui a obtenu le plus de points.

Par exemple, avec la matrice de gains U présentée Figure 1.1, si le premier joueur joue le coup B et le second joue le coup A, alors le premier joueur obtient le gain $U_{21} = 8$ points tandis que le second obtient le gain $U_{12} = 2$ points.

1.2.2 Jeu itéré

Lorsque le jeu est itéré, une partie entre deux joueurs se déroule en l coups successifs. Le nombre l de coups n'est pas connu à l'avance par les joueurs.

A chaque coup, la matrice de gains peut changer. Aussi, les joueurs sont informés de la nouvelle matrice de gains au début de chaque coup. Ils sont également informés, après chaque coup, de leur gain, de celui de leur adversaire, et du coup joué par leur adversaire.

Le gain obtenu par un joueur est égal à la somme des points qu'il a obtenu à chaque coup. Le gagnant est déterminé par le joueur qui a accumulé le plus de points durant la partie.

1.2.3 Tournoi

Un joueur joue selon une stratégie qui décrit complètement son comportement durant le jeu. Afin d'évaluer les stratégies et de pouvoir les comparer, on les fait s'affronter dans un tournoi.

Lorsque p stratégies sont présentes, un tournoi est organisé entre chacune d'elles. Chaque stratégie joue, tour à tour, une partie contre chacune des autres, y compris elle-même.

Afin que toutes les stratégies soient évaluées de manière équitable, les l matrices de gains, utilisées pour les l coups de chaque partie, sont tirées une fois pour toutes au début du tournoi. Ainsi, à chaque rencontre entre deux stratégies, c'est la même i^{eme} matrice qui est utilisée au i^{eme} coup. Les différentes rencontres d'un tournoi sont donc soumises au même environnement.

Le score obtenu par une stratégie est égal à la somme des scores que cette stratégie a obtenus dans chacune des parties jouées. On pourra construire

	S1	S2	S3	Score
S1	510	608	332	1450
S2	312	560	671	1543
S3	405	489	495	1389

FIG. 1.2 – Exemple de matrice de scores d'un tournoi entre 3 stratégies

une matrice de taille $p \times p$ des scores obtenus dans chacune des rencontres.

Par exemple, avec la matrice de scores présentée Figure 1.2 (à laquelle on a ajouté une colonne indiquant le score total pour le tournoi), la stratégie S1 obtient des scores de 510 points en jouant contre elle-même, de 608 points en jouant contre la stratégie S2 et de 332 points en jouant contre S3, soit 1450 points au total. Avec un score de 1543 points au total, c'est la stratégie S2 qui remporte ce tournoi.

1.3 Variantes

Plusieurs variantes de ce jeu sont envisagées en fonction des contraintes que l'on impose sur le tirage des matrices de gains. Ces contraintes peuvent porter à la fois sur la taille de la matrice de gains et sur les valeurs des scores.

1.3.1 Cas général

Dans le cas le plus général, la taille de la matrice de gains est variable et peut prendre n'importe quelle valeur. Dans la pratique, il faut bien fixer une limite, et nous avons donc permis à la taille de la matrice de varier entre 2 et 5. Une matrice de taille 1 n'a que très peu d'intérêt puisqu'il n'y a dans ce cas plus aucun choix à faire, et la taille minimale de 2 est évidente. Pour la taille maximale, une valeur inférieure à 5 aurait induit une variation trop faible, et une valeur supérieure, outre le peu d'apport entre 5 et 6, aurait amené des temps de calculs trop importants. La valeur de 5 nous paraît un bon compromis entre plage de variations de la taille de la matrice et temps de calcul.

En ce qui concerne les valeurs des gains de la matrice, il faut également fixer une limite, c'est à dire choisir la valeur de la variable k. Nous avons choisi k = 10, ce qui signifie que la valeur des gains est tirée aléatoirement entre 0 et 10 inclus. Cette valeur permet d'obtenir une bonne plage de variations. Étant donné la taille des matrices, qui contiennent au plus $5 \times 5 = 25$ valeurs, une limite supérieure pourrait entraîner des écarts trop importants entre les gains et ainsi donner trop d'attraits à certains choix pour qu'il y ait réellement un dilemme. L'autre problème, qu'entraine une valeur plus

importante pour la limite, est l'augmentation de la taille de la variable stockant le score au niveau du simulateur, et donc l'augmentation du temps de calcul.

1.3.2 Matrices binaires

Une des variantes, qu'il est possible de donner au jeu, est de fixer $k = 1$. Ainsi, plutôt que de choisir une valeur arbitraire pour la limite, on choisit de travailler sur des valeurs binaires.

Pour un jeu simple, le joueur gagne s'il a un gain de 1 et son adversaire de 0, il fait match nul si les deux joueurs ont le même gain et il perd s'il a 0 et son adversaire 1.

Pour un jeu itéré, le joueur qui obtient le plus de points est celui qui a emporté le plus de victoires. Cette propriété n'est pas vraie dans le cas général. En contre-exemple, supposons une partie à trois coups dans laquelle le joueur A gagne 10, 0 et 0 points et le joueur B gagne 0, 1 et 1 points. Le joueur A gagne la partie avec 10 points contre 2 points pour B, alors que B a remporté le plus de coups (2 coups victorieux pour B contre un seul pour A).

1.3.3 Matrices de taille fixe

Pour cette autre variante, la taille des matrices est fixe. Cette taille pourra être fixée à 2, 3, 4 ou 5. Au delà, le temps de calcul est trop important.

Des stratégies supplémentaires peuvent être écrites dans le cas de taille de matrice de gains fixe. Par exemple, une stratégie qui calcule la probabilité avec laquelle son adversaire a choisi chacun des coups possibles jusqu'à ce tour, et joue le coup le plus probable. Cette stratégie n'est possible que si le nombre de coups possibles, donc la taille de la matrice de gains, est fixe.

1.3.4 Ensemble exhaustif de matrices

Dans cette variante, les matrices ne sont plus tirées aléatoirement. La partie est jouée, de manière itérée, sur toutes les matrices d'une classe donnée, de manière exhaustive.

Pour que cela soit possible, le nombre de matrices doit être restreint. Ici, les matrices sont à taille fixe, et, de plus, n et k ont obligatoirement un valeur faible.

Le nombre de matrices de dimension n et de valeur dans [0,k] est facile à calculer : il y a k+1 choix possibles pour chaque valeur de la matrice et il y a n^2 valeurs dans la matrice, soit $(k+1)^{n^2}$ matrices.

Ce nombre croit très rapidement, et, en pratique, cette variante ne sera possible que pour k égal à 1 ou 2 (pour n = 3) et 1, 2, 3, 4, 5, 6, 7, 8, 9 ou 10 (pour n = 2).

Chapitre 2

Stratégies

2.1 Présentation

2.1.1 Phases d'une stratégie

Une stratégie décrit totalement le comportement d'un joueur. Elle se décompose en plusieurs opérations qui correspondent à chacune des phases d'une étape d'un jeu. Une étape d'un jeu comprend :

1. une phase d'initiation durant laquelle le joueur est informé de la nouvelle matrice de gains, libre à lui de la stocker ou de la pré-analyser ;
2. une phase de choix durant laquelle le joueur est interrogé sur le coup qu'il désire jouer et doit alors impérativement donner une réponse ;
3. une phase de terminaison durant laquelle le joueur est informé du coup joué par son adversaire, libre à lui de le stocker afin de le prendre en compte pour la prochaine étape du jeu. Cela lui permet également de connaître son score ainsi que celui de son adversaire pour cette étape du jeu.

Une stratégie précise donc quels sont les calculs et décisions à effectuer pour chacune de ces trois phases. Pour les phases d'initiation et de terminaison, il existe un comportement par défaut et le joueur peut ne définir que l'opération de la phase de choix.

Pour l'initiation, le comportement par défaut se contente de stocker la matrice de gains actuelle uniquement pour la durée de cette étape. Pour la terminaison, le comportement par défaut ne fait rien.

Les stratégies définies et utilisées pour les tests sont décrites dans la suite de ce chapitre. Si aucune précision n'est donnée, la description correspond à la phase de choix, et les phases d'initiation et de terminaison utilisent le comportement par défaut.

Ainsi, la stratégie de la première ligne : *"jouer la première ligne de la matrice de gains"* définit une stratégie qui joue toujours le coup numéro 1, la première ligne de la matrice de gains.

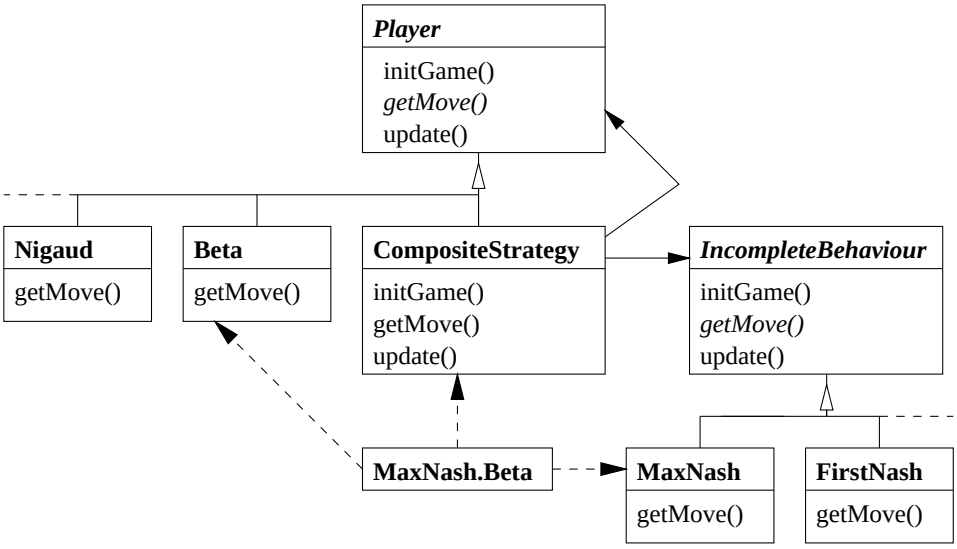

FIG. 2.1 – Schéma de dépendances entre stratégies

2.1.2 Comportement incomplet et stratégies composites

Un comportement incomplet n'est capable de choisir un coup que dans certaines circonstances. Par exemple, le comportement incomplet suivant : *"si la matrice de gains comporte au moins un équilibre de Nash, jouer le premier équilibre de Nash trouvé"*.

Pour définir une stratégie, il faut en plus associer un comportement pour la close "sinon". C'est pourquoi les stratégies composites permettent d'associer un comportement incomplet à une stratégie.

Ainsi, dans notre cas, on peut définir une stratégie composite en disant : *"jouer le comportement incomplet de Nash, sinon, jouer la stratégie de la première ligne"*. Si une stratégie min max est définie, on peut définir une autre stratégie composite : *"jouer le comportement incomplet de Nash, sinon, jouer la stratégie min max"*, sans avoir besoin de redéfinir la méthode de recherche des équilibres de Nash ni celle du calcul du min max.

Ce mécanisme permet de tirer parti des stratégies déjà écrites et d'associer de manière très simple n'importe laquelle de ces stratégies à un comportement incomplet. Si p stratégies ont déjà été écrites, le comportement incomplet de Nash peut être utilisé avec chacune d'elle, créant ainsi p nouvelles stratégies.

La Figure 2.1 représente un schéma de l'implémentation de ce mécanisme. La classe abstraite *IncompleteBehaviour* sert à créer un comportement incomplet, *CompositeStrategy* est une stratégie composite, et *Player* est une classe abstraite servant à créer une stratégie. Le nom de la classe *CompositeStrategy* vient du fait que ce schéma s'inspire du design pattern appelé

Composite et présenté en détails dans [7]. Les méthodes *initGame, getMove* et *update* correspondent, respectivement, aux phases d'initialisation, de choix et de terminaison d'une étape du jeu. Les classes *Nigaud* et *Beta* héritent de *Player* et n'ont que la méthode *getMove* à définir pour être des stratégies, les deux autres méthodes étant celles définies par défaut dans leur classe mère. Les classes *MaxNash* et *FirstNash* héritent de *IncompleteBehaviour* et doivent définir la méthode *getMove* pour être des comportements incomplets, les autres méthodes étant celles définies par défaut. *MaxNash.Beta* est un objet de type *CompositeStrategy* composé du comportement incomplet *MaxNash* et de la stratégie *Beta*.

2.2 Stratégies aveugles

Ces stratégies choisissent une ligne à l'aveugle, c'est à dire sans même prendre en compte les scores de la matrice de gains.

Bêta

Cette stratégie choisit toujours de jouer la première ligne de la matrice de gains.

Remarque : la taille des matrices de gains étant variable, il n'est pas possible de jouer toujours une ligne d'un certain numéro autre que 1 ou 2 (la taille est supérieure à 2).

Nigaud

Cette stratégie choisit toujours de jouer la dernière ligne de la matrice. Elle retourne en fait la taille de la matrice qui peut varier à chaque étape du jeu.

Boucle

Boucle commence par choisir la première ligne, puis la seconde, la troisième et ainsi de suite. Elle incrémente le numéro choisi à chaque étape du jeu.

Quand il n'est pas possible de jouer ce numéro de ligne, c'est à dire quand il est supérieur à la taille de la matrice de gains courante, le numéro est "réinitialisé" à 1.

RandomStrat

RandomStrat joue un numéro de ligne qu'elle choisit aléatoirement entre 1 et la taille de la matrice.

2.3 Stratégies calculatrices

Ces stratégies s'appuient sur les scores de la matrice de gains et vont l'analyser à l'aide de calculs.

Naive

Naïve joue la ligne contenant le gain de valeur maximale. Si plusieurs lignes contiennent un gain de valeur maximale, elle joue la première rencontrée, celle dont le numéro est le plus petit.

SemiNaive

SemiNaïve joue la ligne dont la somme des gains est maximale. Si plusieurs lignes ont une somme maximale, elle joue la première qu'elle rencontre (celle dont le numéro est le plus petit).

Cette stratégie maximise en fait l'espérance du score quand elle joue contre une stratégie jouant aléatoirement. En effet, soient U la matrice de gains de dimension n et p_1 à p_n les probabilités avec lesquelles l'adversaire joue les coups 1 à n. Calculons l'espérance du score obtenu par le joueur dans le cas où il joue le coup i. On a : $E(i) = \sum_j U_{ij} p_j$. En supposant que l'adversaire joue aléatoirement, $p_1 = \ldots = p_n = p$. Ce qui implique : $E(i) = p \sum_j U_{ij}$. Finalement, $\max_i E(i) = p \max_i \sum_j U_{ij}$. De plus, comme la stratégie SemiNaive joue la ligne dont la somme des gains est maximale, elle joue la ligne l telle que $\sum_j U_{lj} = \max_i \sum_j U_{ij}$. Donc, $\max_i E(i) = E(l)$.

SemiNaive maximise bien l'espérance du score quand elle joue contre une stratégie aléatoire. Cependant, la plupart des stratégies joue de manière determinée et ne permettra pas à SemiNaive de maximiser cette espérance. Il faut en fait replacer ce calcul dans le contexte d'un tournoi : les probabilités représentent alors la probabilité qu'une des stratégies de l'échantillon participant au tournoi joue un coup donné. Sur un échantillon contenant toutes sortes de stratégies, pas forcément de bonnes stratégies, on peut penser que les coups sont joués de manière équiprobable. Dans ce cas, SemiNaive maximise à nouveau l'espérance du score, non pas au niveau d'une partie, mais au niveau du tournoi.

Néanmoins, rien n'indique que, dans un échantillon représentatif, les coups soient joués de manière équiprobable. Et même si cette hypothèse est vraie, il peut y avoir des stratégies qui obtiennent un score supérieur au maximum de l'espérance. En réalité, on parle d'espérance si l'on ne connaît pas à l'avance le coup joué par l'adversaire. Si une stratégie est capable de comprendre comment chacune des autres stratégies fonctionne, elle peut prévoir, à coup sur, le coup suivant joué par l'adversaire et ainsi obtenir un score qui sera supérieur au maximum de l'espérance.

AgressiveNaive

Cette stratégie est dite agressive parce qu'elle ne cherche pas à obtenir le meilleur score possible, mais cherche à minimiser celui de l'adversaire.

Le jeu étant symétrique, lorsque le joueur choisit la i^{eme} ligne, son adversaire va obtenir un des gains de la i^{eme} colonne. Minimiser les gains de l'adversaire revient donc à jouer la ligne pour laquelle la colonne correspondante offre les gains les plus faibles.

Cette stratégie calcule les scores maxima de chaque colonne et choisit la ligne symétrique à la colonne pour laquelle cette valeur est la plus basse. Les scores sur les colonnes étant ceux de l'adversaire, cette stratégie assure que le score de l'adversaire ne dépassera pas le minimum des scores maxima sur les colonnes.

AgressiveSemiNaive

Cette stratégie calcule la somme des scores de chacune des colonnes de la matrice des gains. Elle joue ensuite la ligne symétrique à la colonne pour laquelle cette somme est la plus basse.

De même que la stratégie SemiNaïve cherche à maximiser l'espérance du score obtenu par le joueur, AgressiveSemiNaïve cherche à minimiser l'espérance du score obtenu par l'adversaire.

Equitable

Cette stratégie joue la ligne qui a la plus grande valeur sur sa diagonale. Elle est équitable dans le sens qu'elle cherche à ce que son adversaire ait le même score (et soit le plus élevé possible). Ainsi, deux stratégies Équitable en opposition obtiendront les mêmes scores.

Prudente

Cette stratégie calcule les scores minima de chaque ligne et choisit la ligne de plus grand minimum. Elle est prudente au sens où elle s'assure un score minimum le plus élevé possible.

MaxGap

Le jeu étant symétrique, si U est la matrice de gains du joueur, celle de son adversaire est U^t. L'écart entre le gain du joueur et celui de son adversaire est donc donné par la matrice $U - U^t$.

Cette stratégie cherche à maximiser l'écart entre les scores des deux joueurs (en sa faveur bien sur). Elle revient donc à jouer la ligne contenant le maximum pour la matrice $U - U^t$. Dans le cas où il y a plusieurs maxima pour l'écart, elle choisit celui qui associe la plus grande valeur pour le gain du joueur.

Cette stratégie peut être considérée comme agressive puisqu'elle vise à réduire le score de son adversaire en plus d'augmenter son propre score.

MaxLineGap

Tout comme la stratégie MaxGap, cette stratégie vise à maximiser l'écart entre les scores des deux joueurs (toujours en sa faveur). Ici, cependant, on calcule la somme des gains sur les lignes de la matrice $U - U^t$ et on joue la ligne pour laquelle cette somme est maximale.

MaxLineGap cherche à maximiser l'espérance de l'écart de gains entre les deux joueurs.

IntCommunEgoiste

Appelons gain commun la somme du gain obtenu par le joueur et du gain obtenu par son adversaire. On obtient donc une matrice des gains communs en ajoutant à la matrice des gains U sa transposée, donc la matrice $U + U^t$.

Cette stratégie recherche les coups pour lesquels le gain commun est maximal. S'il y a plusieurs maxima, elle joue celui qui offre la plus grande valeur pour son propre gain.

Cette stratégie cherche l'intérêt commun, c'est à dire qu'elle cherche à maximiser le gain commun.

IntCommunLigne

Cette stratégie calcule la somme des gains communs sur chaque ligne de la matrice $U + U^t$ et joue la ligne offrant la somme la plus grande.

Ainsi, elle cherche à maximiser l'espérance du gain commun.

FirstNash

Cette stratégie cherche les équilibres de Nash de la matrice de gains (pour des précisions sur les équilibres de Nash, voir l'annexe A). Elle joue le premier qu'elle rencontre en parcourant la matrice colonne par colonne.

C'est une stratégie incomplète étant donné qu'il n'y a pas toujours un équilibre de Nash, et doit donc être composée avec une autre stratégie.

MaxNash

Cette stratégie cherche les équilibres de Nash de la matrice de gains et joue celui qui offre le gain le plus grand pour le joueur.

C'est une stratégie incomplète.

NashIntCommun

Cette stratégie cherche les équilibres de Nash de la matrice de gains et joue celui qui offre le gain commun le plus grand.

C'est une stratégie incomplète.

2.3.1 Stratégies adaptatives

SelfBest

SelfBest cherche à jouer le mieux possible contre elle-même. Elle vise donc à maximiser ses gains ainsi que ceux de son adversaire quand elle joue contre elle-même. Avec U la matrice de gains, SelfBest va donc essayer de toujours obtenir le score $M = max(U + U^t)$.

Quand M se trouve sur la diagonale, il suffit de jouer la ligne contenant M. Si SelfBest joue contre elle-même, l'adversaire jouera lui aussi cette ligne, et la somme des scores des deux joueurs vaudra M.

Dans le cas où M n'est pas sur la diagonale, il est possible de tirer parti du fait que la matrice $U + U^t$ est symétrique. Il suffit que le premier joueur joue la première ligne (celle de plus petit numéro) contenant M et que le second joueur joue la ligne contenant le symétrique de M. Dans ce cas, on a également la somme des scores des deux joueurs qui vaut M.

Si cette stratégie est possible, alors c'est la meilleure qui joue contre elle-même, comme on l'a expliqué. Dans la pratique, comment peut-on décider que le premier joueur joue la première ligne contenant M et que le second joue son symétrique ? Comment peut-on décider qui est le premier joueur et qui est le second joueur ?

Pour cela, il faut que le joueur s'adapte au comportement de l'adversaire. Ici, il faut que les deux joueurs se synchronisent, qu'ils réussissent à décider qui va jouer la première ligne contenant M et qui va jouer le symétrique, à se mettre d'accord sur leur rôle.

Dans le cas où le premier M est sur la diagonale, il n'y a pas de problème, comme nous l'avons fait remarquer. Dans le cas contraire, les joueurs ont le choix entre deux coups distincts : jouer la première ligne contenant M ou jouer son symétrique. Pour qu'il y ait accord, il faut que les joueurs jouent différemment. Pour y arriver, SelfBest utilise le processus suivant :

- on considère qu'il n'y a pas d'accord avant que le premier coup ne soit joué ;
- tant qu'un accord n'a pas été trouvé, le joueur choisit aléatoirement s'il joue la première ligne ou son symétrique ;
- s'il y a accord, alors on joue de la même manière qu'au coup précédent : première ligne contenant M si c'était le cas, ou son symétrique si c'est le symétrique qui avait été joué au coup précédent ;
- pendant la phase de terminaison, on récupère le coup joué par l'adversaire : si c'est le même que le notre alors il n'y a pas eu accord, s'il

est différent, alors il y a eu accord.

Quand SelfBest joue contre elle-même, il y a quatre situations possibles à chaque coup :

1. les deux joueurs ont joué la première ligne contenant M ;
2. le premier joueur a joué le première ligne et le second joueur a joué le symétrique ;
3. le premier joueur a joué le symétrique et le second joueur a joué la première ligne ;
4. les deux joueurs ont joué le symétrique.

Les situations 2 et 3 conduisent à un accord, les situations 1 et 4 perpétuent le désaccord. Après le premier coup de la partie, il y a donc une probabilité de $\frac{1}{2}$ d'être parvenu à un accord, et une probabilité de $\frac{1}{2}$ d'être encore en désaccord.

De la même manière, après le second coup, il y a une probabilité de $\frac{1}{2}$ de passer d'une situation de désaccord à une situation d'accord. De plus, lorsqu'une situation d'accord a été trouvée, l'accord est conservé pour les coups suivants, les deux joueurs ne modifiant plus leur façon de jouer. La probabilité d'être encore en désaccord après le second coup est donc égale au produit de la probabilité d'être en désaccord après le premier coup et de celle de perpétuer le désaccord pendant le second coup, soit $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$.

Par récurrence, on montre facilement que la probabilité d'être encore en désaccord après le n^{eme} coup est égale à $\frac{1}{2^n}$. Ces calculs ne prennent pas en compte les cas où le premier M est sur la diagonale, mais dans ce cas les deux joueurs jouent le même coup et obtiennent quand même le score maximal. Ces cas ne demandent donc pas d'accord particulier, c'est pourquoi nous ne les avons pas comptés. Il convient donc de reformuler la proposition en : il y a une probabilité de $\frac{1}{2^n}$ pour qu'il y ait plus de n désaccords dans la partie.

En conséquence, il peut très bien ne jamais y avoir d'accord au cours d'une partie. Cependant, la suite des $(\frac{1}{2^n})$ converge rapidement vers 0, et un accord est souvent assez rapidement trouvé. Ainsi, ce processus de mise en accord ou de synchronisation permet à la stratégie SelfBest de se rapprocher très près de la stratégie idéale qui joue le mieux contre elle-même.

Pour plus de détails, voici le code de la stratégie. Dans ce code, la variable booléenne *accord* indique s'il y a accord ou non, la variable booléenne *prioriteVerticale* indique si l'on joue la première ligne contenant M suivant un parcourt vertical de la demi-matrice ou si l'on joue le symétrique. Les variables entières *indiceMaxVert* et *indiceMaxHor* indiquent les indices des premiers maxima rencontrés suivant un parcourt vertical et horizontal respectivement.

```
variables
  accord booleen <- faux;
  prioriteVerticale booleen <- faux;
```

```
  indiceMaxHor entier;
  indiceMaxVert entier;

fonction choixMouvement(out: choix entier)
  debut
     indiceMaxHor <- trouverMaxHor();
     indiceMaxVert <- trouverMaxVert();
     si accord alors
     |  si prioriteVerticale alors choix <- indiceMaxVert
     |  sinon choix <- indiceMaxHor;
     sinon
     |  si indiceMaxHor = indiceMaxVert alors
     |  |  // Le max est sur la diagonale, il suffit de le jouer
     |  |  choix <- indiceMaxHor;
     |  sinon
     |  |  // On cherche un accord en modifiant la priorite
     |  |  prioriteVerticale <- tirageAleatoire(vrai,faux);
     |  |  si prioriteVerticale alors choix <- indiceMaxVert
     |  |  sinon choix <- indiceMaxHor;
  fin

fonction terminaison(in: sonChoix entier)
  debut
     si (!accord et indiceMaxHor != indiceMaxVert) alors
     |  // On ne peut se mettre d'accord sur une priorite
     |  // de jeu que si le max n'est pas sur la diagonale
     |  si (prioriteVerticale et sonChoix = indiceMaxHor)
     |  alors accord <- vrai;
     |  sinon
     |  |  si (!prioriteVerticale et sonChoix = indiceMaxVert)
     |  |  alors accord <- vrai;
  fin
```

SelfBestIB

SelfBestIB est basée sur la stratégie SelfBest et en reprend le principe, à savoir jouer le mieux possible contre soi-même. Cependant, elle corrige le défaut principal de cette stratégie. En effet, alors que SelfBest joue très bien contre elle-même, elle joue de manière inefficace contre les autres joueurs, et obtient de piètres résultats en tournoi.

Pour remédier à ce problème, SelfBestIB va essayer d'identifier son adversaire. Tant que son adversaire joue un maximum de la matrice des gains communs $U + U^t$, donc de manière analogue à la stratégie SelfBest, SelfBestIB va jouer suivant la stratégie SelfBest. Au premier écart de l'adversaire

par rapport au comportement attendu, SelfBestIB passe la main à une autre stratégie.

Ceci est facilité par le fait que SelfBestIB est un comportement incomplet. Il doit donc être associé à une autre stratégie, cette dernière prenant le relais quand l'adversaire ne joue pas suivant la stratégie SelfBest.

Par exemple, si l'on associe SelfBestIB à SemiNaive, le joueur va commencer par utiliser la stratégie SelfBest, puis dès qu'il se rend compte que son adversaire ne joue pas suivant une stratégie SelfBest, il joue suivant la stratégie SemiNaive. Ainsi, on profite à la fois de la stratégie SelfBest, qui joue bien contre elle-même, et de la stratégie SemiNaive, qui maximise l'espérance du gain, contre les autres joueurs.

Le code de SelfBestIB reprend l'intégralité du code de SelfBest. Seuls les ajouts sont indiqués, les partie identiques étant représentées par [../..]. La variable *stratAlternative*, de type *strategie* représente, pour des raisons pratiques, un objet, bien que ce soit en contradiction avec l'approche de description de code utilisée précédemment. C'est un lien vers la stratégie à utiliser dans le cas où l'on ne joue pas contre une autre stratégie SelfBest.

```
variables
  [../..]
  stratAlternative strategie;
  stratInconnue booleen <- faux;

fonction choixMouvement(out: choix entier)
  debut
    si stratInconnue alors
    |  // On passe la main a la strategie alternative
    |  choix <- stratAlternative.choixMouvement();
    sinon
    |  // On reprend le fonctionnement normal de SelfBest
    [../..]
  fin

fonction terminaison(in: sonChoix entier)
  debut
    si stratInconnue alors
    |  // On passe la main a la strategie alternative
    |  stratAlternative.terminaison(sonChoix);
    sinon
    |  si accord alors
    |  |  // Verifie si je joue bien contre SelfBest
    |  |  si (prioriteVerticale et sonChoix != indiceMaxHor) alors
    |  |  |  stratInconnue <- vrai;
    |  |  si (!prioriteVerticale et sonChoix != indiceMaxVert) alors
```

```
  |  |  |  stratInconnue <- vrai;
  |  sinon
  |  |  si (indiceMaxHor != indiceMaxVert) alors
  |  |  |  // Est-on tombe d'accord ?
  |  |  |  si (prioriteVerticale et sonChoix = indiceMaxHor) alors
  |  |  |  |  accord <- vrai;
  |  |  |  si (!prioriteVerticale et sonChoix = indiceMaxVert) alors
  |  |  |  |  accord <- vrai;
  |  |  // Verifie si je joue bien contre SelfBest
  |  |  si (sonChoix != indiceMaxHor et sonChoix != indiceMaxVert) alors
  |  |  |  stratInconnue <- vrai;
fin
```

Chapitre 3

Expérimentations

3.1 Jeu du cas général

Dans cette partie, le jeu est celui du cas général présenté dans le chapitre précédent. Une partie se déroule donc sur 1000 coups, les matrices de gains sont tirées de manière aléatoire, de taille entre 2 et 5, et à valeur dans [0,10].

Afin de pouvoir reproduire les tests à l'identique, le jeu de matrices de gains est toujours le même, résultat d'un tirage avec la clé 960021934641 dans le simulateur.

3.1.1 Stratégies aveugles et calculatrices

Les stratégies aveugles sont difficilement départageables : leurs scores sont trop proches. Par contre, parmi les stratégies calculatrices, la stratégie SemiNaive se détache, comme le montrent les résultats du tounoi et de l'évolution, illustrés figure 3.1 et 3.2 respectivement.

De manière assez prévisible, il y a un écart notable au niveau des scores du tournoi entre, d'une part, les stratégies calculatrices, et d'autre part les stratégies aveugles et les stratégies agressives. En effet, les stratégies aveugles n'analysent pas les matrices de gains et obtiennent des scores autour de la moyenne. Quant aux stratégies agressives, elles minimisent à la fois leurs scores et ceux de leurs adversaires, donc leurs scores en tournoi sont faibles.

En évolution, on aperçoit également la coupure entre les deux groupes de stratégies. Par ailleurs, SemiNaive domine largement l'évolution, prenant le dessus dès le départ, et finissant seule au détriment de toutes les autres stratégies.

3.1.2 Stratégies avec équilibres de Nash

L'annexe B a mis en évidence qu'une grande proportion des matrices de taille entre 2 et 5 ont au moins un équilibre de Nash. Il est donc intéressant d'ajouter aux stratégies précédentes celles basées sur les équilibres de Nash.

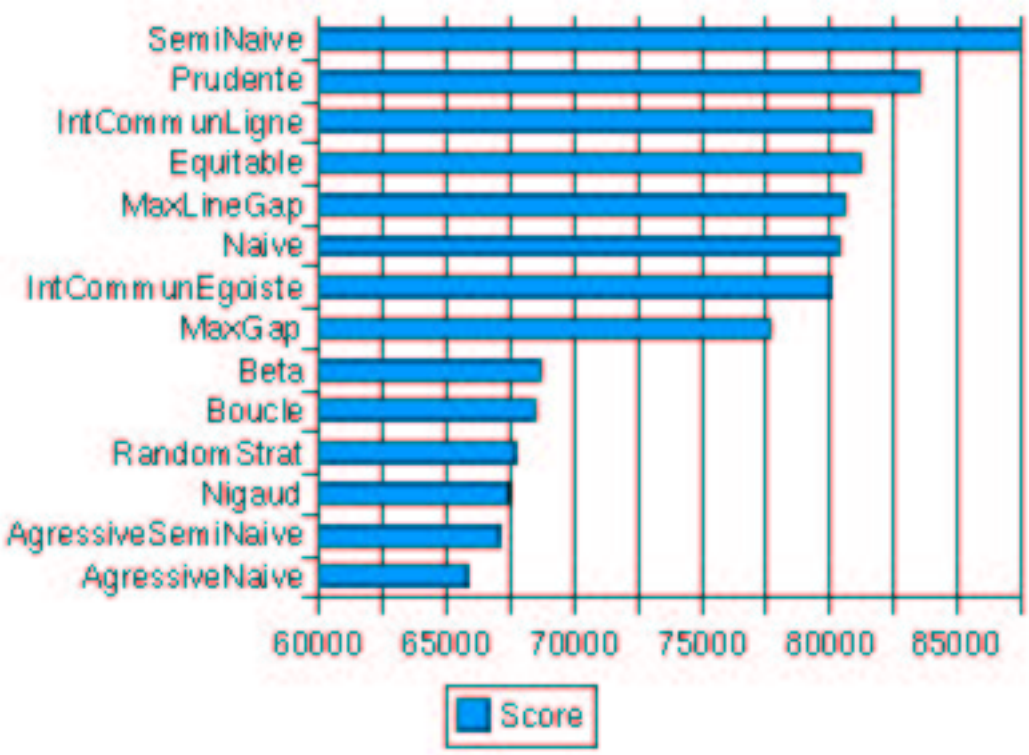

FIG. 3.1 – Scores en tournoi dans le cas général entre stratégies aveugles et calculatrices

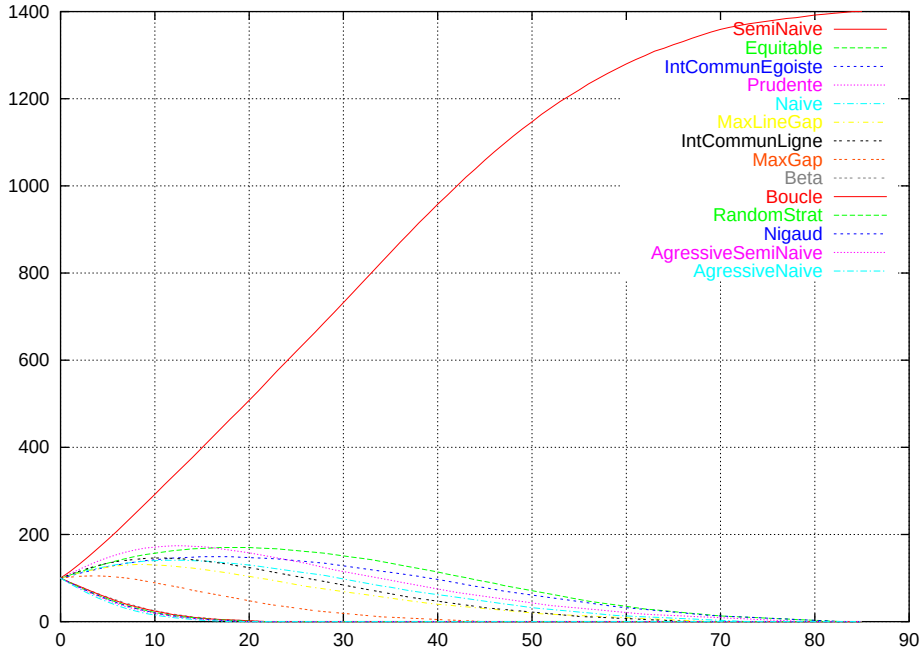

FIG. 3.2 – Evolution entre stratégies aveugles et calculatrices dans le cas général

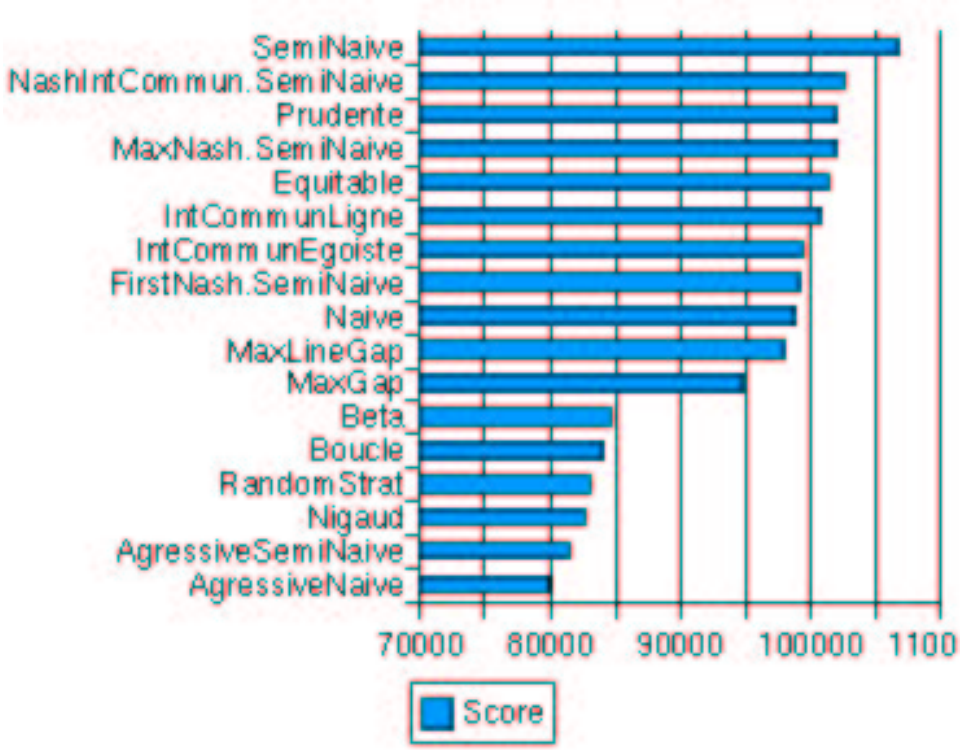

FIG. 3.3 – Tournoi avec stratégies basées sur les équilibres de Nash dans le cas général

Pour leur permettre d'obtenir de bons scores dans le cas où il n'y a pas d'équilibre de Nash, c'est la stratégie SemiNaive, qui avait donné les meilleurs résultats précédemment, qui sera utilisée en complément.

Les figures 3.3 et 3.4 présentent les résultats obtenus lors du tournoi et lors de l'évolution respectivement.

En ce qui concerne le tournoi, il y assez peu de changements : la coupure entre les deux groupes de stratégies est toujours bien visible, et SemiNaive se détache à nouveau en tête. On peut remarquer que les stratégies utilisant les équilibres de Nash se comportent assez bien : NashIntCommun se classe deuxième, MaxNash quatrième, et FirstNash huitième.

Pour l'évolution, par contre, la situation a changé : même si SemiNaive se comporte très bien au départ, elle disparaît, canibalisée par la stratégie Equitable. Seule NashIntCommun réussit à lui résister. Ceci prouve que l'ajout d'une stratégie peut influencer les résultats des autres et bouleverser les classements, surtout en évolution. Pour pallier à ce problème, nous faisons appel aux sous-classes de stratégies.

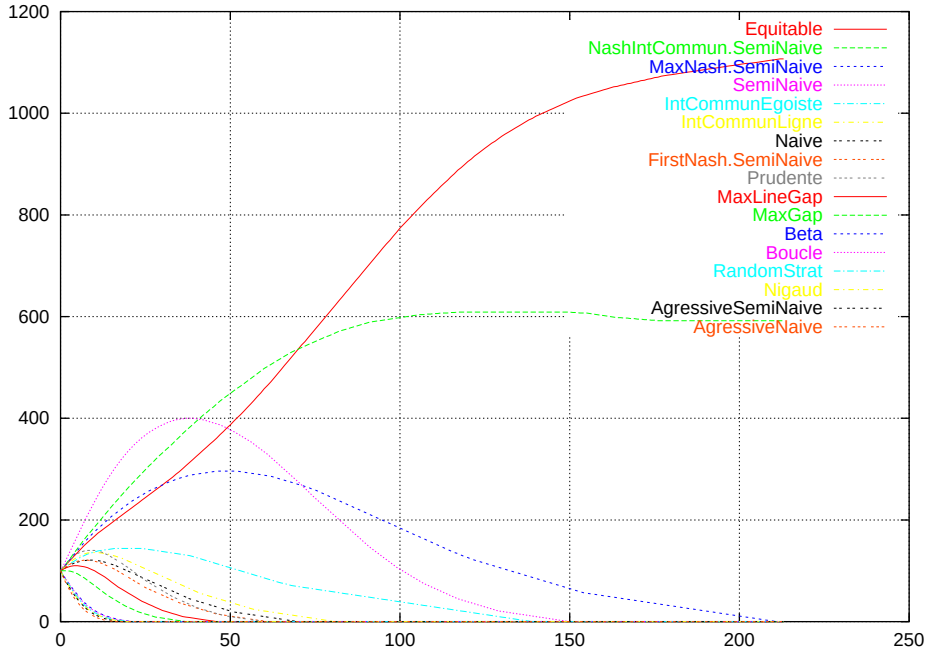

FIG. 3.4 – Evolution avec stratégies basées sur les équilibres de Nash dans le cas général

3.1.3 Sous-classes de stratégies

En considérant un ensemble de n stratégies comme une classe, on effectue les tests (tournoi et évolution) sur tous les sous-ensembles (sous-classes) de $n - k$ stratégies, avec k fixé. En prenant par exemple $k = 1$, l'influence d'une stratégie sur les résultats est réduite. Les résultats obtenus sont plus robustes.

En pratique, pour chaque sous-classe, un tournoi et une évolution sont éffectués. Pour chaque stratégie, on compte le nombre de tournois et d'évolutions qu'elle gagne.

Au total, il y a C_n^{n-k} sous-classes, mais une stratégie donnée n'apparaît pas dans toutes les sous-classes, et ne peut donc pas gagner toutes les compétitions. Plus précisément, une stratégie apparaît dans C_{n-1}^{n-k-1} sous-classes. Les résultats seront donc donnés, pour chaque stratégie, en pourcentage de compétitions gagnées par rapport au nombre de compétitions auxquelles elle a participé.

En conséquence, la somme des scores pourra dépasser 100% : par exemple, si la stratégie A gagne toutes ses compétitions, soit 100%, et que la stratégie B gagne toutes les compétitions auxquelles A n'a pas participé, par exemple 4%. La somme des scores atteint alors 104%.

Des tests ont été effectués avec la classe de stratégies précédente et différentes valeurs pour k. Les résultats sont présentés figures 3.5 pour les tournois et 3.6 pour les évolutions.

La stratégie SemiNaive a gagné tous les tournois auxquels elle a participé, elle peut être considérée comme la meilleure en tournoi parmi toutes les stratégies précédentes. En seconde place viennent NashIntCommun et Prudente.

En évolution, la stratégie Equitable s'impose la majeure partie du temps. Cependant, pour les sous-classes avec $k = 4$ (sous-ensembles à 13 stratégies parmi les 17 de départ), elle perd plus de 20% de ses évolutions, au bénéfice de SemiNaive.

3.1.4 Stratégie SelfBest

La stratégie SelfBest présentée au chapitre précédent est la stratégie qui obtient les meilleurs scores en jouant contre elle-même. Elle devrait donc obtenir de bons résultats en évolution, et c'est ce que nous allons vérifier.

La figure 3.7 présente les scores obtenus en tournoi. SemiNaive tient toujours la tête du classement et SelfBest obtient une honorable seconde place.

La figure 3.8 présente l'évolution entre toutes les stratégies. Le résultat prévu et attendu est bien atteint : SelfBest a dominé toutes les autres stratégies en évolution.

Des tests sur les sous-classes de stratégies ont été menés pour vérifier la

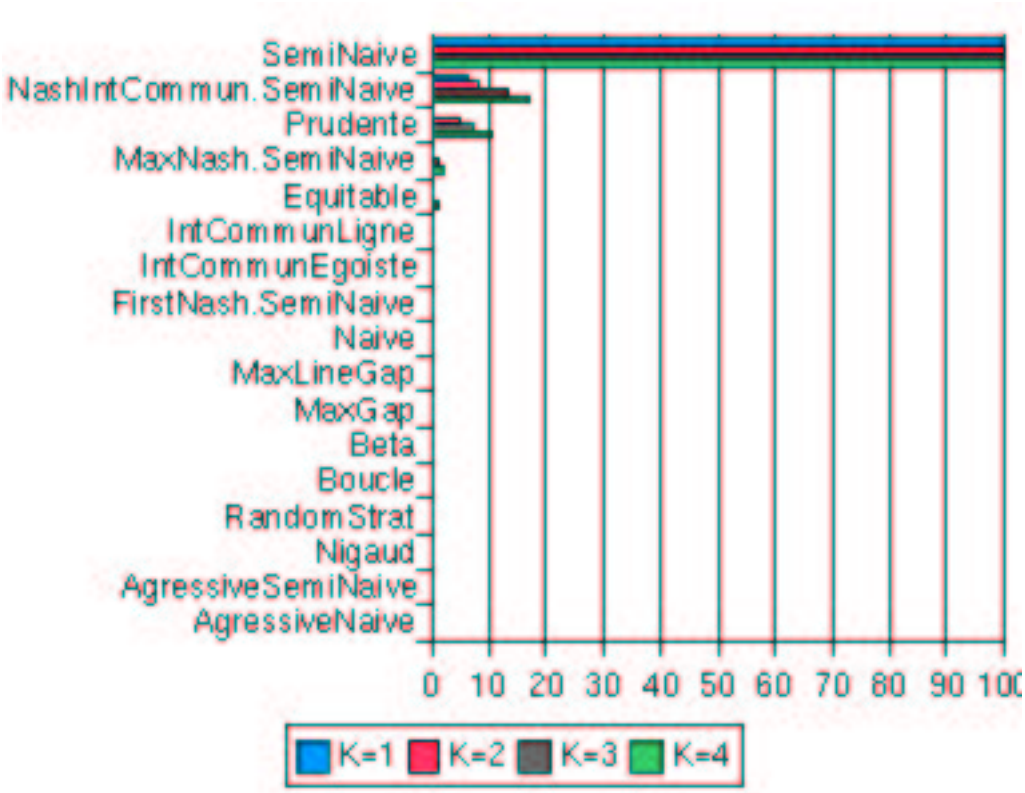

FIG. 3.5 – Pourcentages de tournois gagnés en sous-classes de stratégies

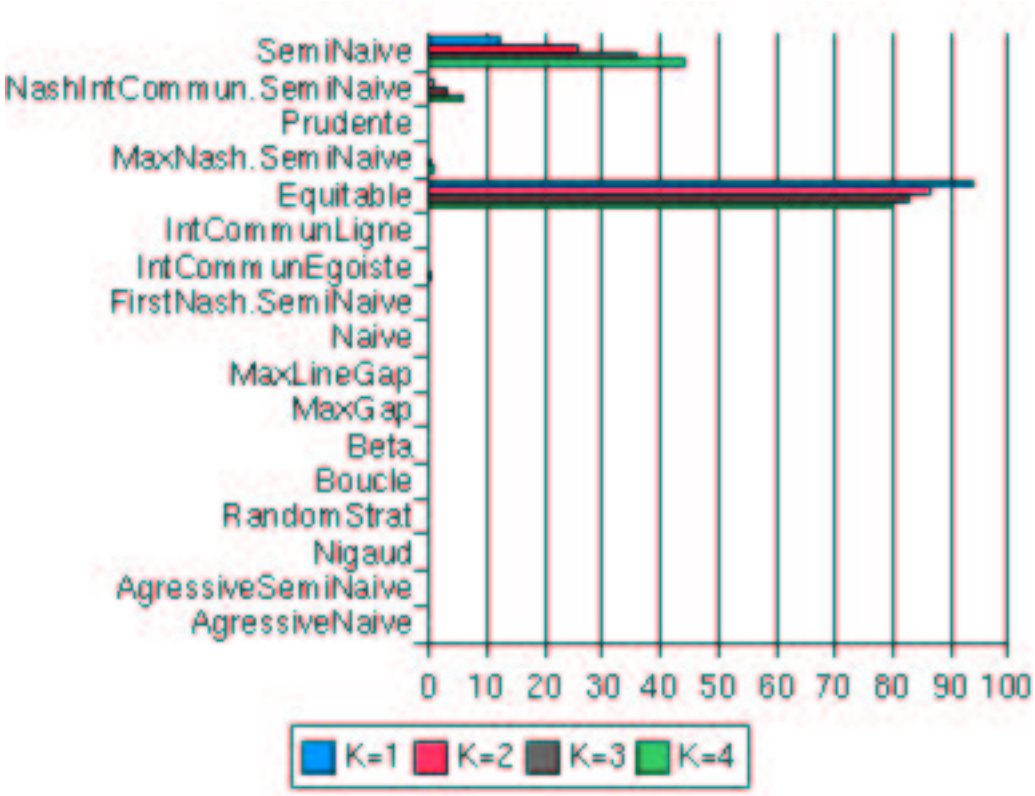

FIG. 3.6 – Pourcentages d'évolutions gagnées en sous-classes de stratégies

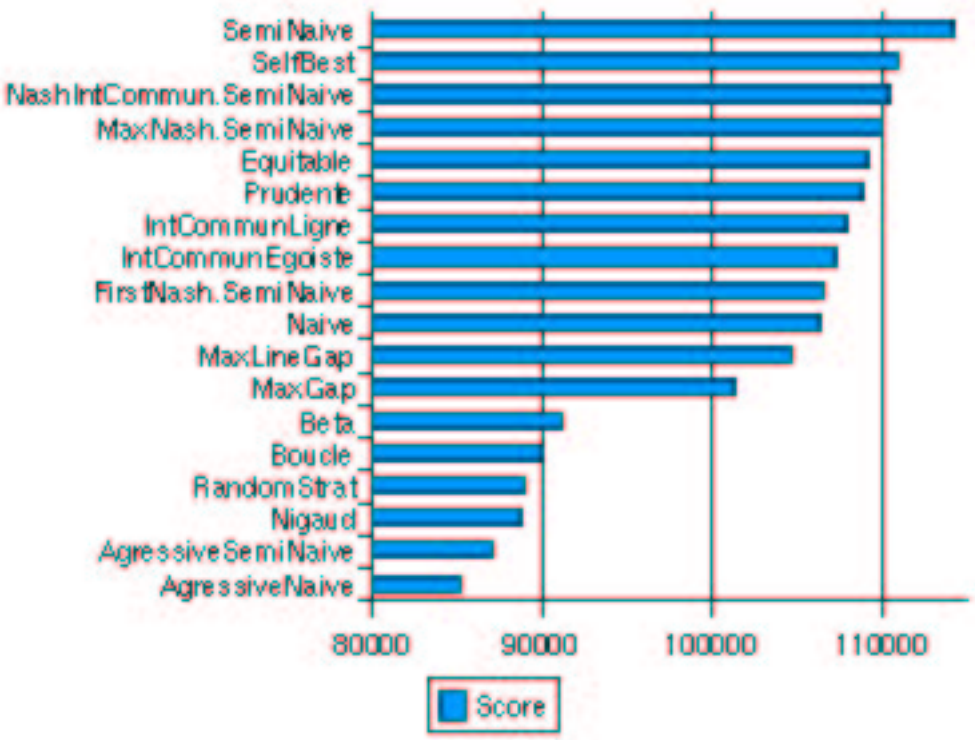

FIG. 3.7 – Scores en tournoi avec SelfBest

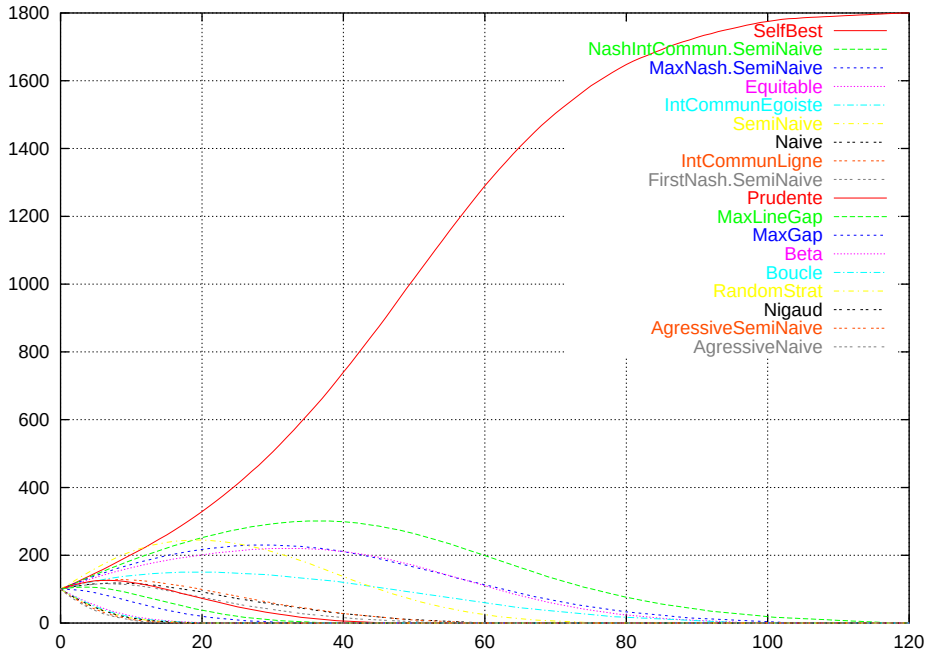

FIG. 3.8 – Evolution avec la stratégie SelfBest

robustesse des résultats. SemiNaive a gagné tous ses tournois, sauf un peu moins de 1% pour $k = 4$. Quant à SelfBest, elle a gagné toutes ses évolutions, sans exception. En conséquence, les graphiques concernant les sous-classes ont assez peu d'intérêt et n'ont pas été inclus.

3.1.5 La stratégie SelfBestIB.SemiNaive

SelfBest est sans conteste la meilleure stratégie, parmi les précédentes, en évolution, et SemiNaive la meilleure en tournoi. Une nouvelle stratégie a été créée pour bénéficier à la fois des bonnes propriétés de SemiNaive en tournoi et des bonnes propriétés de SelfBest en évolution.

La stratégie SelfBestIB.SemiNaive se comporte en effet comme SelfBest quand elle joue contre elle-même et comme SemiNaive quand elle joue contre une autre stratégie.

La figure 3.9 présente les résultats en tournoi. SelfBestIB.SemiNaive l'emporte d'une courte longueur sur SemiNaive, comme on s'y attendait puiqu'elle joue la plupart du temps comme SemiNaive, sauf contre elle-même, cas où elle obtient un meilleur score que SemiNaive.

La figure 3.10 présente l'évolution de l'ensemble des stratégies. SelfBestIB.SemiNaive conserve bien les bonnes propriétés de SelfBest et domine l'évolution du début jusqu'à la fin.

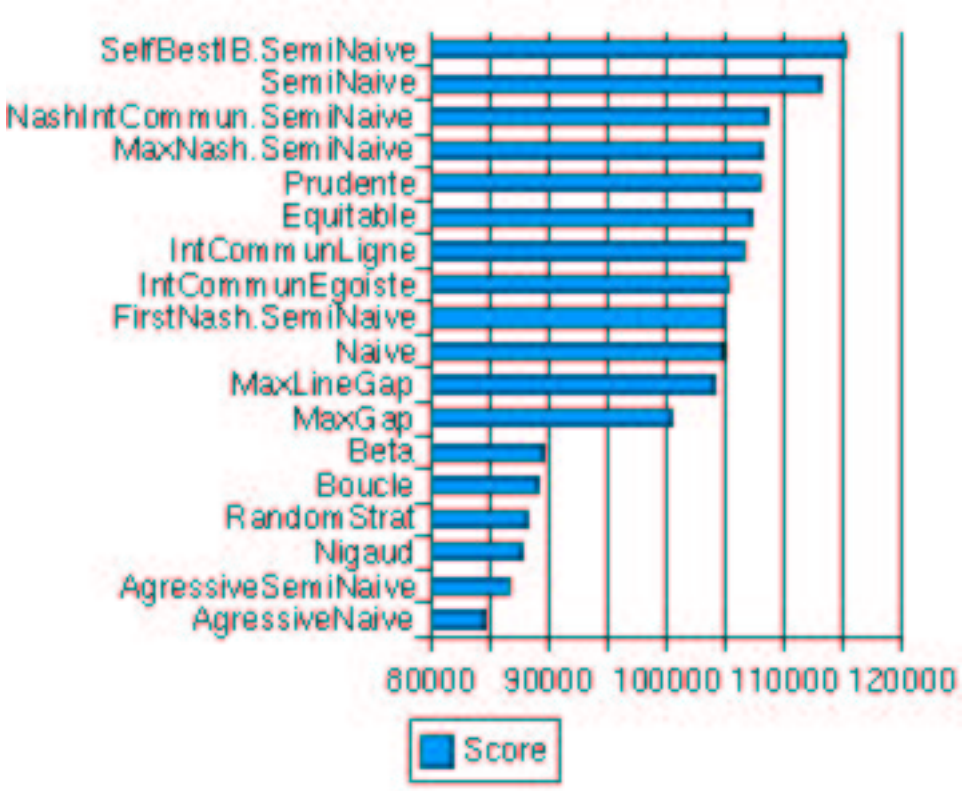

Fig. 3.9 – Scores en tournoi avec SelfBestIB.SemiNaive

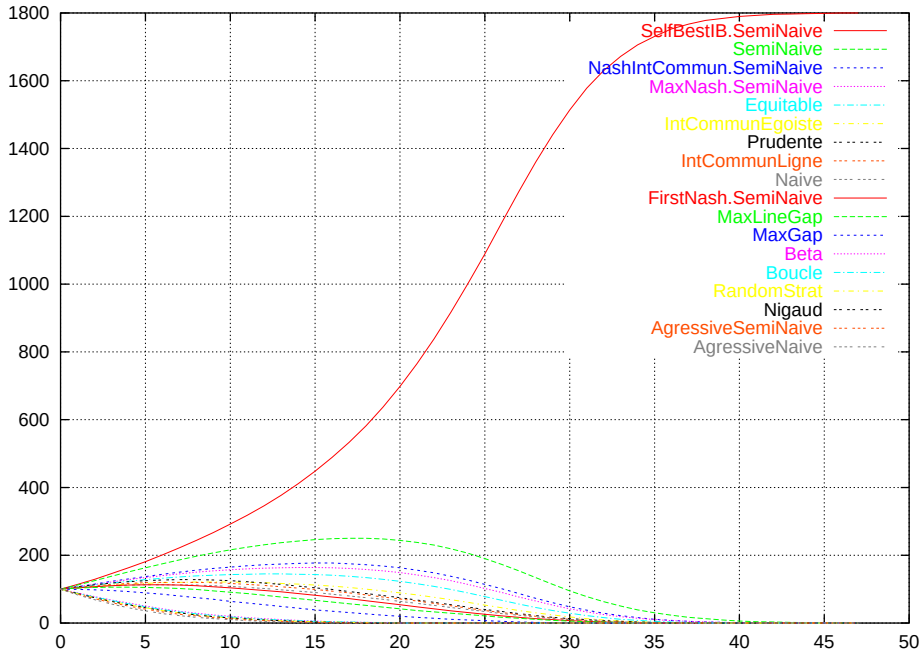

FIG. 3.10 – Evolution avec la stratégie SelfBestIB.SemiNaive

Les tests sur les sous-classes de stratégies confirment ses résultats : pour k égal à 1, 2, 3 et 4, SelfBestIB.SemiNaive a gagné tous ses tournois et toutes ses évolutions, sans exception. Cette nouvelle stratégie combine donc parfaitement les bonnes caractéristiques de SelfBest et de SemiNaive.

3.2 Matrices de gains avec un et un seul équilibre de Nash

La section précédente a permis de mettre en évidence les performances relatives des différentes stratégies. Nous testons maintenant les stratégies dans une version légèrement différente du jeu : toutes les matrices de gains doivent contenir un et un seul équilibre de Nash. Le but est de montrer, si c'est possible, que jouer l'équilibre de Nash n'est pas la meilleure stratégie, et donc de trouver une stratégie qui obtient de meilleurs scores en tournoi et gagne les évolutions.

Etant donné qu'il n'y a qu'un seul équilibre de Nash, les trois stratégies FirstNash, MaxNash et NashIntCommun (qui jouent respectivement le premier équilibre de Nash trouvé, celui de gain maximum et celui qui maximise le gain commun) vont toujours jouer de la même manière. Les tests reprennent donc l'ensemble des stratégies utilisées dans les tests précédents

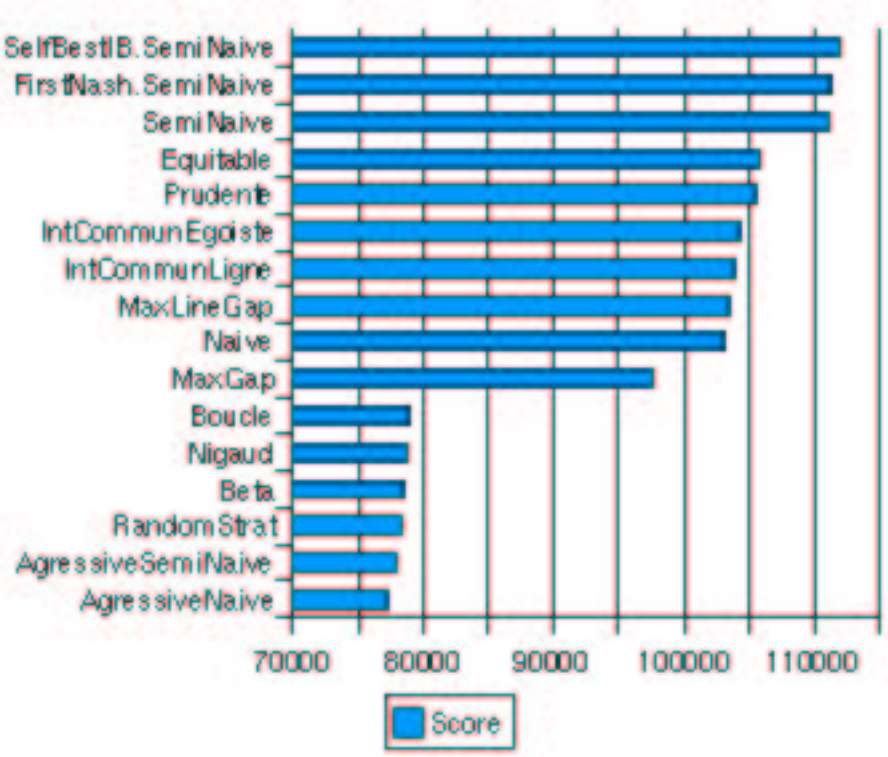

FIG. 3.11 – Scores en tournoi, variante avec un et un seul équilibre de Nash

mais avec une seule de ces trois stratégies, ici FirstNash.

La figure 3.11 illustre les scores obtenus en tournoi. La stratégie SelfBestIB.SemiNaive l'emporte devant FirstNash, mais de justesse. La stratégie qui consiste à jouer l'équilibre de Nash obtient un très bon score, meilleur que celui de la stratégie SemiNaive.

L'évolution illustrée figure 3.12 est remportée par SelfBestIB.SemiNaive. FirstNash arrive à nouveau en seconde position, devançant également SemiNaive.

En ce qui concerne les sous-classes de stratégies, SelfBestIB.SemiNaive gagne 100% des tournois et évolutions, à l'exception de moins d'un pourcent de tournois pour $k = 4$.

FirstNash semble donc être battue. Néanmoins, en étudiant les scores en tournoi, on s'aperçoit que SemiNaive domine les scores au niveau des stratégies aveugles, alors que FirstNash domine au niveau des stratégies calculatrices. Il faudrait faire des tests sur un échantillon plus grand de stratégies pour confirmer les résultats.

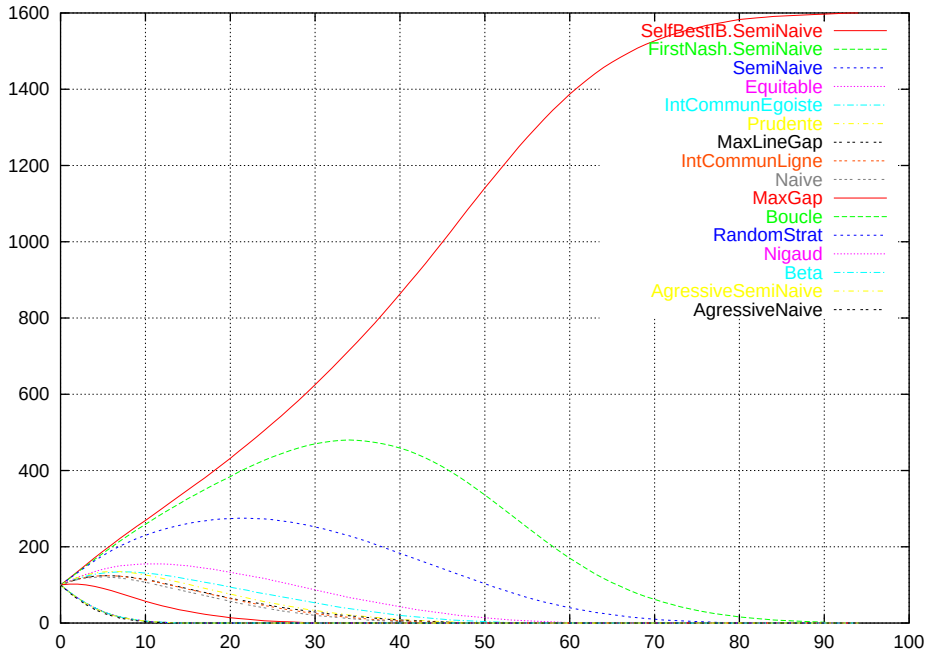

FIG. 3.12 – Evolution, variante avec un et un seul équilibre de Nash

3.3 Matrices de gains avec au moins un équilibre de Nash

Dans cette variante du jeu, les matrices de gains contiennent toutes au moins un équilibre de Nash. Ainsi, les trois stratégies FirstNash, MaxNash et NashIntCommun vont avoir des comportements différents. Les tests porteront donc sur un ensemble de stratégies un peu plus grand.

Les résultats en tournoi, illustrés figure 3.13, sont assez proches de ceux obtenus dans le cas général : SelfBestIB.SemiNaive obtient le meilleur score, suivie de près par SemiNaive. Entre les trois stratégies tirant partie des équilibres de Nash, NashIntCommun s'en tire le mieux, suivie de MaxNash puis FirstNash.

L'évolution, illustrée figure 3.14, est remportée assez largement par SelfBestIB.SemiNaive, suivie de SemiNaive. NashIntCommun et MaxNash sont respectivement troisième et quatrième.

SelfBestIB.SemiNaive gagne également 100% de ses tournois et évolutions en sous-classes de stratégies.

Finalement, les résultats obtenus dans cette variante sont très proches de ceux obtenus dans le cas général, en dehors d'une légère amélioration pour NashIntCommun et MaxNash. Ceci était prévisible, car, comme nous l'avons fait remarqué, une majorité des matrices de gains contiennent au

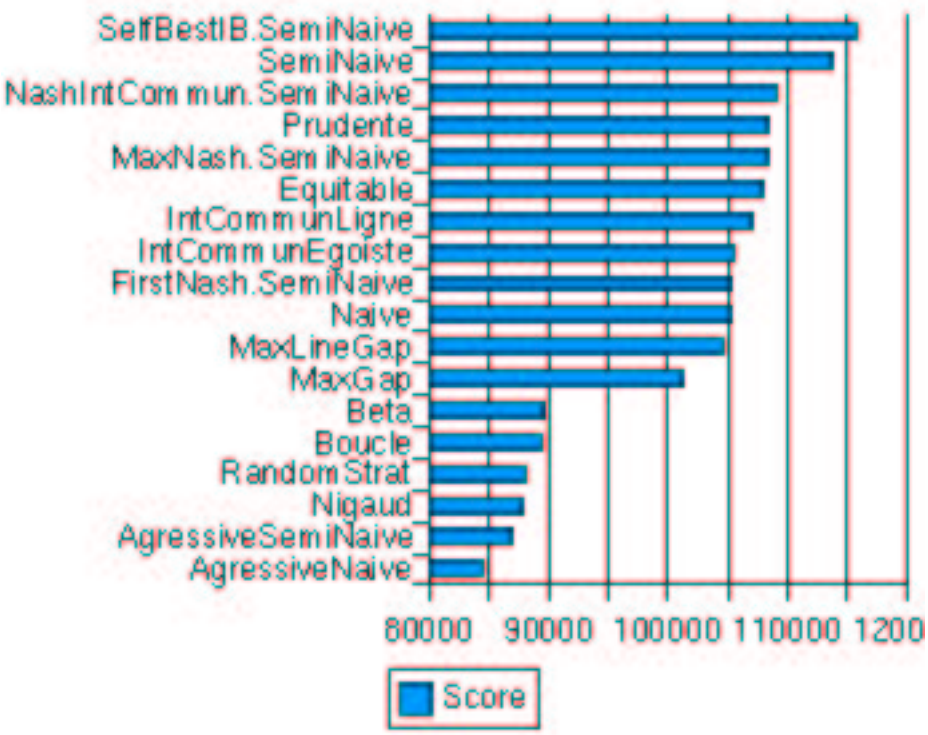

FIG. 3.13 – Scores en tournoi, variante avec au moins un équilibre de Nash

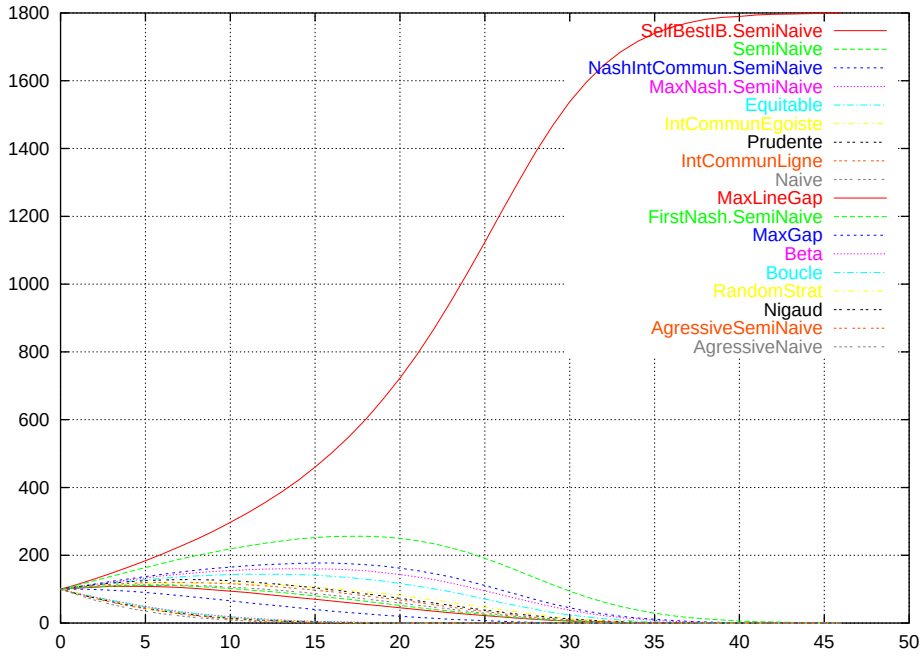

FIG. 3.14 – Evolution, variante avec au moins un équilibre de Nash

moins un équilibre de Nash.

3.4 Matrices binaires

Dans cette variante, les gains des matrices sont binaires (0 ou 1). La figure 3.15, qui illustre les scores obtenus en tournoi, montre encore une fois la domination de SelfBestIB.SemiNaive et de SemiNaive. L'écart est même un peu plus marqué que dans le cas général.

L'évolution, illustrée figure 3.16, est à nouveau remportée par SelfBestIB.SemiNaive, suivie de SemiNaive.

Lors des tests sur les sous-classes de stratégies, SelfBestIB.SemiNaive a gagné tous ses tournois et toutes ses évolutions, sans aucune exception.

Finalement, les résultats obtenus avec les matrices de gains binaires sont proches de ceux obtenus dans le cas général, avec une hiérarchie un peu plus marquée entre les stratégies. Ceci est rassurant et prouve une certaine stabilité des résultats. Seul le cas où les matrices de gains ont un et un seul équilibre de Nash les met en doute par le fait que SemiNaive est dominée par la stratégie jouant l'équilibre de Nash. Cependant, même dans ce cas, la stratégie SelfBestIB.SemiNaive domine toutes les autres, tant en tournoi qu'en évolution.

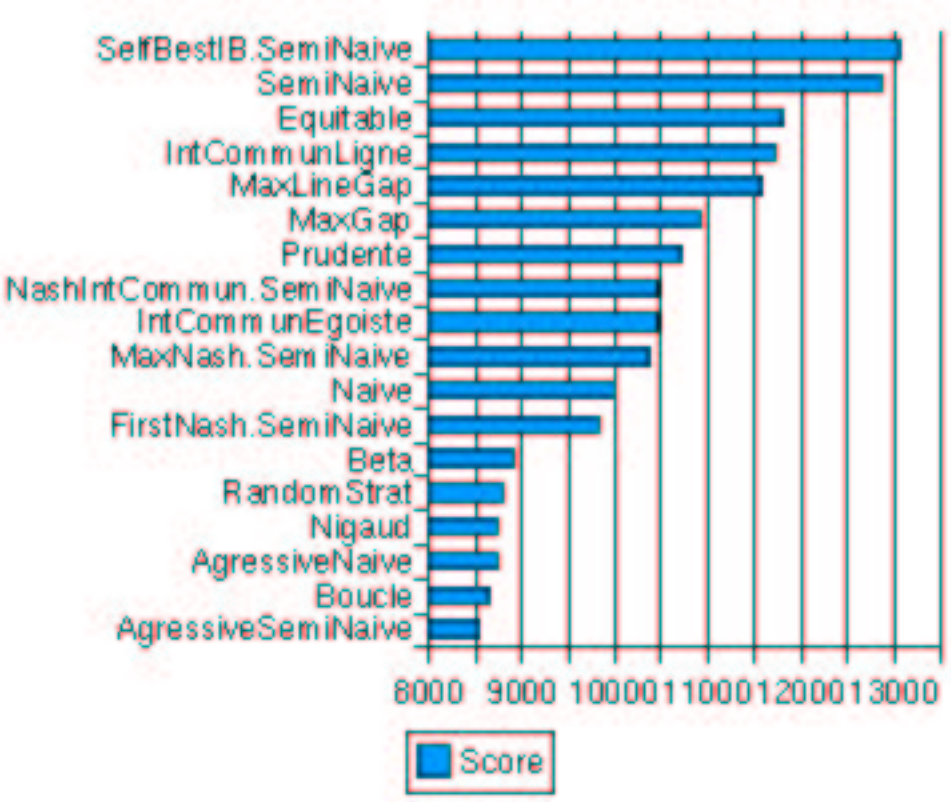

FIG. 3.15 – Scores en tournoi, variante avec matrices de gains binaires

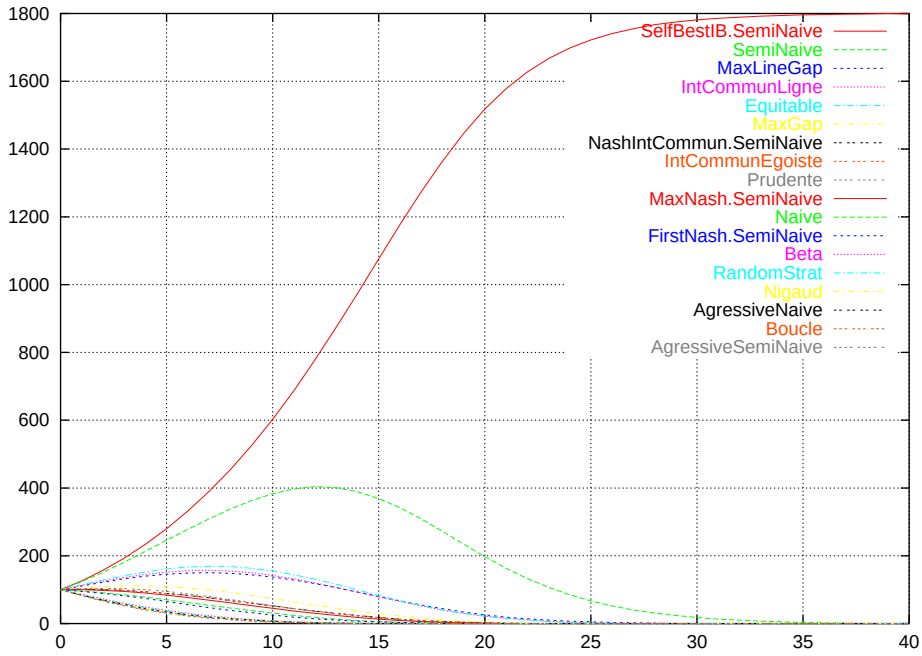

FIG. 3.16 – Evolution, variante avec matrice de gains binaires

Chapitre 4

Résultats

4.1 Caractéristiques d'une bonne stratégie

Les expérimentations ont permis de mettre à jour plusieurs résultats. D'abord, il y a un écart entre deux groupes de stratégies comme ceci apparaît sur les histogrammes des tournois. D'un côté un groupe qui obtient des résultats faibles. Il est constitué des stratégies aveugles et des stratégies agressives. De l'autre côté, un groupe qui obtient d'assez bons résultats. Il est composé des stratégies calculatrices non agressives.

Comme l'on pouvait s'y attendre, les stratégies aveugles, qui n'analysent pas la matrice de gains, obtiennent de faibles résultats. La première caractéristique d'une bonne stratégie est donc qu'elle analyse la situation. Elle doit prendre en compte la matrice de gains, l'analyser et repérer quels coups peuvent lui rapporter le plus de points. La grande différence par rapport au dilemme du prisonnier, c'est qu'il n'y a plus de choix simple entre coopérer ou trahir : les stratégies doivent elles-mêmes donner un sens aux différents coups possibles.

Les stratégies agressives, qui cherchent uniquement à diminuer les points de l'adversaire, otiennent de piètres résultats, comme cela était le cas dans le dilemme du prisonnier. Même si les stratégies agressives gagnent certaines de leur parties, elles obtiennent des scores très faibles et finissent en dernières positions en tournoi et en évolution. Une bonne stratégie ne doit donc pas forcément chercher à gagner une partie contre son adversaire, mais doit chercher à obtenir le score le plus élevé possible, quitte à perdre sa partie. Elle peut perdre une bataille pour gagner la guerre.

4.2 La stratégie qui domine

Les résultats obtenus en tests lors des expérimentations doivent être confirmés sur un jeu de stratégies plus important. Néanmoins, nous avons conçu, en plusieurs étapes, une stratégie qui obtient de très bons résultats.

En effet, SelfBestIB.SemiNaive a gagné tous ses matchs, que ce soit en tournoi ou en évolution. Ses très bons résultats se sont avérés robustes : elle domine également lors des tests effectués sur les sous-classes de stratégies. De plus, elle domine toujours les tests effectués sur les différentes variantes du jeu.

SelfBestIB.SemiNaive peut donc être considérée comme la meilleure des stratégies qui ont été écrites. Elle met en évidence une autre caractéristique d'une bonne stratégie, notamment en évolution : la nécessité de bien jouer contre soi-même.

4.3 Equilibres de Nash

Dans le cas où toutes les matrices de gains ont un et un seul équilibre de Nash, même si la stratégie jouant l'équilibre obtient de bons résultats, elle est dominée par SelfBestIB.SemiNaive.

Dans le cas où les matrices de gains ont au moins un équilibre de Nash, les stratégies jouant l'équilibre sont dominées de manière plus marquée, et par deux autres stratégies. Bien que ceci ne constitue pas réellement une preuve, ce sont des cas où il y a toujours un équilibre de Nash, et où il existe une meilleure stratégie que celle qui joue l'équilibre.

Les tests montrent donc bien que jouer l'équilibre de Nash n'est pas forcément la meilleure stratégie. C'est un des résultats que nous cherchions à obtenir durant ce projet. En effet, il est très intéressant, car jouer un équilibre de Nash est souvent considéré comme la meilleure façon de jouer, comme le coup raisonnable.

De manière intuitive, ceci s'explique par le fait qu'il n'est pas possible de réellement jouer un équilibre de Nash. En pratique, on joue une ligne qui contient un équilibre de Nash et le fait qu'on l'atteigne dépend du coup joué par l'adversaire.

4.4 Influence du choix des stratégies

4.4.1 Sous-classes de stratégies

Toutes les stratégies ont été écrites au sein de la même équipe et peuvent donc être biaisées. Prenons par exemple le cas de la section précédente. Il faut signaler qu'il est possible d'écrire une stratégie qui tire partie du fait que son adversaire joue un équilibre de Nash et l'empêche de l'atteindre, voire qui minimise les points de son adversaire dans ce cas. Soulignons le fait que ceci ne nous intéresse pas et peut être génant pour plusieurs raisons :

- une telle stratégie minimiserait les points obtenus par les stratégies jouant un équilibre de Nash mais pas les points des autres stratégies ;
- cette stratégie obtiendrait certainement un faible score.

Les stratégies écrites ne doivent pas tirer partie du fait que telle ou telle stratégie est présente. Ceci créerait des relations entre les stratégies et biaiseraient les résultats. En effet, ce qui nous intéresse, ce sont les performances des stratégies dans le cas le plus général possible.

Afin de minimiser l'influence du choix des stratégies, les tests ont également été effectués sur les sous-classes de stratégies. Supposons deux stratégies A et B, la stratégie A étant écrite spécialement pour permettre à B d'obtenir de bons résultats. Si A et B font partie de l'échantillon pour le tournoi et l'évolution écologique, A va artificiellement améliorer les résultats de B. L'utilisation de sous-classes de stratégies, par exemple, pour un échantillon de 12 stratégies, tous les sous-ensembles de 10 stratégies, va donner lieu à plusieurs tournois dans lesquels la stratégie B sera présente et pas la stratégie A. Ainsi, l'influence de A sur les résultats de B sera amoindrie.

En ce qui concerne les résultats précédents, ils se sont révélés robustes, puisqu'ils sont vrais également sur les sous-classes.

4.4.2 Génotype

Néanmoins, le nombre de stratégies est trop faible pour espérer être représentatif du cas général, c'est à dire un échantillon représentatif de l'ensemble des stratégies possibles.

Il serait de ce fait intéressant de décrire les stratégies à l'aide d'un génotype et d'un phénotype, comme cela avait été proposé dans [1] pour les stratégies du dilemme du prisonnier. Le génotype est le code représentant la stratégie et le phénotype la signification du génotype.

L'utilisation d'un génotype offre plusieurs avantages, comme cela a été le cas pour le dilemme du prisonnier : la génération automatique d'un grand nombre de stratégies, leur simplicité d'expression et la possibilité de les dénombrer facilement. Ainsi, des personnes néophytes en programmation peuvent proposer des stratégies et ceci permettrait de créer un concours en ligne où chacun pourrait saisir une stratégie et la voir s'affronter contre toutes les autres. La création d'un tel concours est également un des buts de ce projet, car il permettrait, dans le cas d'une participation importante, d'obtenir un échantillon de stratégies relativement représentatif.

Le génotype utilisé pour le dilemme du prisonnier n'est pas directement réutilisable et nous n'avons malheureusement pas réussi à l'adapter aux méta-jeux. Les seules adaptations imaginées étaient trop limitées. Ceci est d'ailleurs un inconvénient majeur de l'utilisation d'un génotype : la limitation dans l'expression des stratégies.

4.4.3 Choix par éliminations successives

Principe

Les génotypes imaginés ne permettant pas de décrire des stratégies moyennement évoluées, il est peut-être possible de les décrire comme une suite de traitements. L'idée est de sélectionner le comportement désiré par éliminations successives sur les combinaisons de comportements (une combinaison de comportements est l'association du comportement du joueur et de celui de son adversaire).

Au départ, aucun choix n'est fait, toutes les combinaisons de comportements sont possibles. Il y aurait ainsi une liste comprenant toutes les combinaisons de comportements, c'est à dire toutes les cases de la matrice de gains.

Ensuite, le joueur procède à une série de filtres de sélection. Par exemple, les filtres "Ne conserver que les combinaisons qui m'offrent le gain maximum parmi les combinaisons qui n'ont pas encore été éliminées" ou "S'il y a au moins un équilibre de Nash parmi les combinaisons restantes, ne conserver que celles qui représentent un équilibre de Nash".

La liste des combinaisons va donc diminuer au fur et à mesure. Les filtres doivent assurer qu'ils ne vident pas la liste car le joueur doit donner obligatoirement un choix.

A la fin, s'il reste plusieurs combinaisons dans la liste, on choisit la première. Enfin, le choix du joueur est le comportement associé à la combinaison choisie.

Intérêt et exemples

Pour que cette manière d'opérer fonctionne, il faut définir un grand nombre de filtres afin de donner aux utilisateurs la plus grande liberté possible dans la création de leur stratégie. Une stratégie se décrit alors comme une suite ordonnée de filtres. Sa description est donc simple et permet de générer un grand nombre de stratégies en changeant l'ordre des filtres.

Cette méthode reprend les bonnes caractéristiques du génotype et permet en plus de décrire des stratégies évoluées. Par exemple, prenons les filtres :

- MAX "Ne conserver que les combinaisons qui m'offrent un score maximal" ;
- DIAG "Ne conserver que les combinaisons où mon comportement est le même que celui de mon adversaire s'il y en a au moins une, sinon les conserver toutes" ;
- NASH "Ne conserver que les combinaisons représentant un équilibre de Nash s'il y en a au moins un, sinon les conserver toutes".

La stratégie Naive est équivalente au filtre MAX, Equitable à la suite DIAG-MAX, FirstNash au filtre NASH, MaxNash.Naive à la suite NASH-

MAX. La description est donc très simple tout en permettant de créer des stratégies évoluées.

Cette méthode n'a pas été réalisée faute de temps. L'idée peut tout de même servir de base pour un développement ultérieur, et peut-être pour lancer un concours en ligne. Suivant les retours des utilisateurs, de nouveaux filtres peuvent être ajoutés, ce qui rend cette solution finalement assez souple, extensible et simple d'utilisation.

Conclusion

Les méta-jeux représentent une bonne extension du dilemme du prisonnier : la dynamicité des règles, les comportements variables et plus nombreux permettent d'appréhender un plus grand nombre de situations d'interactions et de s'approcher un peu plus de la réalité.

Différents résultats ont été mis à jour. Certains sont identiques à ceux du dilemme du prisonnier : la nécessité d'obtenir de bons scores avant même de gagner une partie, la nécessité de jouer le mieux possible contre soi-même pour s'imposer en évolution. D'autres sont plus spécifiques : la nécessité d'analyser la matrice de gains, de donner soi-même un sens aux différents coups possibles.

L'écriture des stratégies a mis à jour d'autres résultats. Ainsi, jouer les équilibres de Nash ne donne pas les meilleurs résultats bien que ce soit considéré comme la façon raisonnable de jouer. La stratégie SelfBestIB met en évidence également une certaine complexité pour obtenir de bons résultats. Cette complexité se retrouve également dans la recherche d'un moyen simple d'écriture de stratégies qui permettrait de réaliser un site de concours en ligne.

Annexe A

Recherche des équilibres de Nash

A.1 Contexte

Le contexte est celui décrit précédemment : on considère donc un jeu à deux joueurs qui jouent simultanément 1 coup parmi n. Le jeu est à somme non nulle et les scores sont déterminés par une matrice de gains de taille $n \times n$. Cette matrice contient les scores qu'un joueur obtient contre son adversaire en fonction des deux coups joués.

Dans ce jeu, les valeurs de la matrice de gains sont comprises entre 0 et k. Cette matrice et la variable k sont connues des deux joueurs et le gagnant est celui qui a obtenu le plus de points.

On se propose ici de décrire un algorithme permettant de localiser les équilibres de Nash présents dans la matrice de gain, s'il y en a.

A.2 Définition

Comme défini dans [1]

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

On parle également de "situation de non regret" : au vu des choix des autres joueurs, aucun joueur ne regrette son choix stratégique.

A.3 Analyse

Dans le cas de joueurs A et B ayant, respectivement, le choix entre des stratégies $a_1, \ldots, a_n$ et $b_1, \ldots, b_m$, le couple de stratégies (a_i, b_j) est un équilibre de Nash si et seulement si :

1. sachant que B joue la stratégie b_j, A ne regrette pas d'avoir joué la stratégie a_i
2. sachant que A joue la stratégie a_i, B ne regrette pas d'avoir joué la stratégie b_j

Plaçons nous dans un jeu non symétrique. Soient U et V les matrices des gains obtenus, respectivement, par A et B. Dans le cas du choix du couple de stratégies (a_i, b_j), A obtient le gain U_{ij} et B le gain V_{ij}.

Proposition 1 *(a_i, b_j) est un équilibre de Nash si et seulement si $U_{ij} = \max_k U_{kj}$ et $V_{ij} = \max_k V_{ik}$.*

Preuve
L'existence des maxima ne fait aucun doute puisqu'il y a un nombre fini (non nul) de stratégies pour chacun des deux joueurs. On montre donc les deux sens de l'équivalence.

Soit (a_i, b_j) un équilibre de Nash. Alors, connaissant la stratégie b_j de B, le joueur A ne regrette pas d'avoir joué la stratégie a_i. Donc, toute stratégie a_k jouée par A lui aurait amené un gain U_{kj} plus faible (au sens large). C'est à dire : $\forall k \in \{1, \ldots, n\}, U_{kj} \leq U_{ij}$, donc $U_{ij} = \max_k U_{kj}$.
De même, en considérant le choix du joueur B, on obtient $V_{ij} = \max_k V_{ik}$.

Réciproquement, supposons que $U_{ij} = \max_k U_{kj}$ et $V_{ij} = \max_k V_{ik}$.
Alors, $\forall k \in \{1, \ldots, n\}, U_{kj} \leq U_{ij}$. C'est à dire, pour une stratégie b_j donnée du joueur B, la stratégie a_i est celle qui maximise les gains de A.
On a aussi $\forall k \in \{1, \ldots, m\}, V_{ik} \leq V_{ij}$. Pour une stratégie a_i donnée du joueur A, la stratégie b_j est celle qui maximise les gains de B.
Finalement, (a_i, b_j) est un équilibre de Nash.

D'après la proposition, les équilibres de Nash sont les couples tels que le gain de A maximise sa colonne (dans la matrice des gains) et le gain de B maximise sa ligne.

Algorithme de recherche : il suffit de chercher les maxima sur les colonnes de la matrice U et les maxima sur les lignes de la matrice V. Les couples (U_{ij}, V_{ij}) pour lesquels U_{ij} et V_{ij} sont parmi les maxima sont les équilibres de Nash.

A.4 Exemple

Illustrons la méthode de recherche par un exemple sur un jeu dont la matrice de gains est illustrée figure A.1.

Effaçons tous les gains de U (resp. V) qui ne sont pas maximum sur leur colonne (resp. ligne). Il reste donc la matrice illustrée figure A.2.

Il y a donc deux équilibres de Nash : les couples (a_1, b_1) et (a_2, b_2).

U/V	b_1	b_2	b_3
a_1	5/4	2/2	0/1
a_2	2/0	4/2	3/2
a_3	0/2	3/5	5/1

FIG. A.1 – Exemple de matrice gains d'un jeu non symétrique

U/V	b_1	b_2	b_3
a_1	5/4	/	/
a_2	/	4/2	/2
a_3	/	/5	5/

FIG. A.2 – Matrice de gains avec maxima de ligne ou de colonne seuls

A.5 Jeux symétriques

Dans un jeu symétrique, les joueurs jouent dans les mêmes conditions. Ainsi, ils ont le choix entre les mêmes stratégies, que nous nommerons $s_1, \ldots, s_n$, et ont les mêmes gains.

Par conséquence, il n'y a qu'une matrice de gains U. Pour un couple de stratégies (s_i, s_j), le joueur A obtient un gain U_{ij} et le joueur B un gain U_{ji}.

Proposition 2 *Dans un jeu symétrique, le couple de stratégies (s_i, s_j) est un équilibre de Nash si et seulement si $U_{ij} = \max_k U_{kj}$ et $U_{ji} = \max_k U_{ki}$.*

Preuve
On peut se ramener à un jeu non symétrique en ajoutant une matrice de gain V pour le joueur B avec $\forall i, j, V_{ij} = U_{ji}$, donc $V = U^t$.
De plus, on sait, d'après la proposition 1, que (s_i, s_j) est un équilibre si et seulement si $U_{ij} = \max_k U_{kj}$ et $V_{ij} = \max_k V_{ik}$. Avec $V = U^t$, ceci revient à $U_{ij} = \max_k U_{kj}$ et $U_{ji} = \max_k U_{ki}$.

Au final, (s_i, s_j) est un équilibre si et seulement si U_{ij} et U_{ji} sont maximum sur leur colonne (resp. colonnes j et i).

A.6 Algorithme de recherche des équilibres dans un jeu symétrique

L'algorithme proposé utilise la proposition 2. Il consiste à isoler les maxima des colonnes de la matrice de gains. Ensuite, pour savoir si le couple (s_i, s_j) est un équilibre, il suffit de vérifier que U_{ij} et U_{ji} sont parmi les maxima.

L'algorithme est décrit avec U la matrice de gains et M une matrice telle que :

- $M_{ij} = 1$ si U_{ij} est un maximum sur la colonne j ;
- $M_{ij} = 0$ sinon.

```
fonction trouverMaximaColonnes(in: U matrice, in: n entier, out: M
matrice)
  variables
    max entier;
    i,j entier;

  debut
     pour j de 1 a n faire
     |  max <- 0;
     |  pour i de 1 a n faire
     |  |  M[i][j] <- 0;
     |  |  si U[i][j] > max alors max <- U[i][j];
     |  pour i de 1 a n faire
     |  |  si U[i][j] = max alors M[i][j] <- 1;
  fin
```

Cet algorithme est de complexité $O(n^2)$, avec n la taille de la matrice de gains U.

```
fonction estEquilibreNash(in: M matrice, in: i entier, in: j entier,
out: reponse booleen)
  debut
     si M[i][j] = 1 et M[j][i] = 1 alors reponse <- vrai;
     sinon reponse <- faux;
  fin
```

Cette fonction est de complexité $O(1)$.

Proposition 3 *Dans un jeu symétrique, il y a équivalence entre :*

(s_i, s_j) *est un équilibre de Nash et*

(s_j, s_i) *est un équilibre de Nash.*

Preuve

D'après la proposition 2, on sait que :

(s_i, s_j) est une équilibre de Nash

$\Leftrightarrow U_{ij} = \max_k U_{kj}$ et $U_{ji} = \max_k U_{ki}$

$\Leftrightarrow (s_j, s_i)$ est une équilibre de Nash.

Si l'on cherche tous les équilibres de Nash, on doit parcourir la matrice complète (la moitié si l'on prend en compte la proposition 3), ce qui demande un appel à la fonction `trouverMaximaColonnes` suivi de n^2 (ou $n^2/2$) appels à la fonction `estEquilibreNash`. Finalement, la recherche de l'ensemble des équilibres de Nash est de complexité $O(n^2)$.

Annexe B

Dénombrement des équilibres de Nash

B.1 Présentation

Dans la section 1.3.4, nous présentons des ensembles exhaustifs de matrices de dimension n fixe et dont les gains sont compris entre 0 et k. Afin de les utiliser dans les tests, il est intéressant de les étudier et de connaître le nombre d'équilibres de Nash de ces matrices.

Dans ce but, un dénombrement de matrices en fonction du nombre d'équilibres qu'elles contiennent a été effectué sur les classes dont la taille reste raisonnable. Dans certains cas, on a pu généraliser leur nombre pour k quelconque.

B.2 Dénombrement

Les figures B.1, B.2 et B.3, présentent les résultats du dénombrement. Nous avons déjà vu, dans la section 1.3.4, que la classe de dimension n et

Nb Nash	0	1	2	3	4	Total
k = 1	0	2	6	4	4	16
k = 2	0	18	36	18	9	81
k = 3	0	72	120	48	16	256
k = 4	0	200	300	100	25	625
k = 5	0	450	630	180	36	1296
k	0				$(k+1)^2$	$(k+1)^4$

FIG. B.1 – Nombre de matrices de dimension 2 et à valeurs dans [0,k] en fonction du nombre d'équilibres de Nash

Nb Nash	0	1	2	3	4	5	6	7
k = 1	2	21	66	102	108	93	64	36
k = 2	250	2025	4740	5324	3798	2142	972	324
k = 3	5488	38808	78960	73232	40416	17496	5952	1440
k								

Nb Nash	8	9	Total
k = 1	12	8	512
k = 2	81	27	19683
k = 3	288	64	262144
k		$(k+1)^3$	$(k+1)^9$

FIG. B.2 – Nombre de matrices de dimension 3 et à valeurs dans [0,k] en fonction du nombre d'équilibres de Nash

Nb Nash	0	1	2	3	4	5	6	7	8
k = 1	122	876	2844	5808	8688	10380	10444	9052	6900

Nb Nash	9	10	11	12	13	14	15	16	Total
k = 1	4720	2840	1572	778	320	144	32	16	65536
k								$(k+1)^4$	$(k+1)^{16}$

FIG. B.3 – Nombre de matrices de dimension 4 et à valeurs dans [0,k] en fonction du nombre d'équilibres de Nash

de valeurs dans $[0, k]$ comprend $(k+1)^{n^2}$ matrices.

Les tableaux indiquent en supplément le nombre de matrices ayant un nombre d'équilibres de Nash donné. Par exemple, dans la classe des matrices de dimension 2 et de valeurs dans $[0, 2]$, il y a 18 matrices ayant un seul équilibre de Nash, 36 en ont 2, 18 en ont 3 et 9 en ont 4.

B.3 Cas des matrices binaires de dimension 2

Ce cas comprend un faible nombre de matrices, 16 exactement. Il est donc possible de toutes les présenter, ce qui permet de se faire une bonne idée du problème.

La figure B.4 présente l'ensemble de ces matrices, en indiquant pour chacune ses équilibres de Nash. Pour chaque matrice, nous indiquons le nombre d'équilibres de Nash qu'elle contient, et les équilibres sont indiqués en gras sur la matrice elle-même.

B.4 Résultats

Suite à ce dénombrement, plusieurs résultats ont été mis à jour. Nous rappelons pour ces résultats que les jeux traités sont symétriques.

U	A	B
A	**0**	**0**
B	**0**	**0**

(1) 4 équilibres

U	A	B
A	**1**	0
B	0	**0**

(9) 2 équilibres

U	A	B
A	**0**	0
B	0	**1**

(2) 2 équilibres

U	A	B
A	**1**	0
B	0	**1**

(10) 2 équilibres

U	A	B
A	0	**0**
B	**1**	**0**

(3) 3 équilibres

U	A	B
A	**1**	**0**
B	**1**	**0**

(11) 4 équilibres

U	A	B
A	0	0
B	1	**1**

(4) 1 équilibre

U	A	B
A	**1**	0
B	1	**1**

(12) 2 équilibres

U	A	B
A	**0**	**1**
B	**0**	0

(5) 3 équilibres

U	A	B
A	**1**	1
B	0	0

(13) 1 équilibre

U	A	B
A	**0**	**1**
B	**0**	**1**

(6) 4 équilibres

U	A	B
A	**1**	1
B	0	**1**

(14) 2 équilibres

U	A	B
A	0	**1**
B	**1**	0

(7) 2 équilibres

U	A	B
A	**1**	**1**
B	**1**	0

(15) 3 équilibres

U	A	B
A	0	**1**
B	**1**	**1**

(8) 3 équilibres

U	A	B
A	**1**	**1**
B	**1**	**1**

(16) 4 équilibres

FIG. B.4 – Nombre d'équilibres de Nash dans les matrices binaires de dimension 2

Proposition 4 *Toutes les matrices de gain de dimension 2 ont au moins un équilibre de Nash.*

Preuve
D'après le proposition 2, U_{ij} correspond à un équilibre de Nash si et seulement si $U_{ij} = \max_k U_{kj}$ et $U_{ji} = \max_k U_{ki}$.
Or, chacune des 2 colonnes de la matrice de gains contient au moins un gain de valeur maximale pour sa colonne. Nous allons étudier les placements possibles du maximum de la première colonne.

Si le maximum de la première colonne est U_{11}, alors on a $U_{11} = \max_k U_{k1}$, et comme U_{11} est son propre symétrique, cela suffit, d'après la proposition 2, à montrer que U_{11} est un équilibre de Nash.

Sinon, le maximum de la première colonne est U_{21}. Observons alors le placement du maximum de la seconde colonne. Soit U_{12} est maximum de la deuxième colonne, on a alors $U_{21} = \max_k U_{k1}$ et $U_{12} = \max_k U_{k2}$, ce qui d'après la proposition montre que U_{21} est un équilibre de Nash (et U_{12} est aussi un équilibre de Nash). Soit U_{22} est maximum de la deuxième colonne. Dans ce cas, on a $U_{22} = \max_k U_{k2}$, et comme U_{22} est son propre symétrique, cela suffit à montrer que U_{22} est un équilibre de Nash.

Au total, on a toujours un équilibre de Nash.

Proposition 5 *Il y a $(k+1)^n$ matrices de dimension n et de valeur dans [0,k] ayant n^2 équilibres de Nash.*

Preuve
U est une matrice de gain de dimension n et de valeur dans [0,k] ayant n^2 équilibres de Nash
$\Leftrightarrow \forall i, j \in [1, n]^2$, U_{ij} est un équilibre de Nash
$\Leftrightarrow \forall i, j \in [1, n]^2$, $U_{ij} = \max_p U_{pj}$ et $U_{ji} = \max_p U_{pi}$
$\Leftrightarrow \forall i, j \in [1, n]^2$, $U_{ij} = \max_p U_{pj}$
$\Leftrightarrow \forall j \in [1, n]$, on a $\forall i \in [1, n]$, $U_{ij} = \max_p U_{pj}$
Ce qui veut dire que toutes les valeurs d'une même colonne sont identiques. Il suffit de dénombrer les matrices ayant cette propriété. Pour la première colonne, on peut avoir tous les gains égaux à 0, tous à 1, ..., tous à k. De même pour les $n-1$ colonnes restantes. Il y a donc k+1 possibilités pour chaque colonne, de manière indépendante, soit $(k+1)^n$ matrices.

Proposition 6 *Il y a $\frac{(k+1)^n kn}{2}$ matrices de dimension n et de valeur dans [0,k] ayant $n^2 - 1$ équilibres de Nash.*

Preuve
Soit U une matrice de gain de dimension n, de valeur dans [0,k] ayant $n^2 - 1$ équilibres de Nash. Ceci veut dire que toutes les valeurs de U correspondent à des équilibres de Nash, sauf une.

La seule valeur qui n'est pas équilibre de Nash est forcément sur la diagonale de U. Dans le cas contraire, son symétrique ne serait pas non plus un équilibre de Nash (d'après la proposition 3), et il y aurait deux valeurs ne correspondant pas à un équilibre de Nash.

En tirant parti de la preuve de la proposition précédente, chaque colonne a toutes ses valeurs identiques, sauf une colonne pour laquelle la valeur de la diagonale est distincte. La valeur de la diagonale est donc inférieure strictement à la valeur maximale de la colonne.

Il suffit de dénombrer les cas possibles d'une telle propriété. Supposons que la valeur non équilibre se trouve sur la première colonne, et donc en U_{11}. On a alors $\forall i \in [2, n]$, $U_{i1} = \max_p U_{p1}$ et $U_{11} < \max_p U_{p1}$. Si le maximum vaut k, U_{11} peut valoir 0, 1, ... ou k-1, soit k valeurs possibles. Si le maximum vaut k-1, U_{11} peut valoir 0, 1, ... ou k-2, soit k-1 valeurs possibles. Pour former la première colonne, nous avons $k + k - 1 + ... + 1 = \frac{(k+1)k}{2}$ possibilités. Pour les n-1 colonnes restantes, il y a $(k+1)^{n-1}$ possibilités, comme nous l'avons montré dans la preuve de la proposition précédente. Au total, il y a $\frac{(k+1)k}{2} \times (k+1)^{n-1} = \frac{(k+1)^n k}{2}$ matrices avec la valeur non équilibre sur la première colonne.

Il y a le même nombre de matrices avec la valeur non équilibre sur la deuxième colonne, la troisième, etc. Finalement, il y a $\frac{(k+1)^n k}{2} \times n = \frac{(k+1)^n kn}{2}$ matrices ayant $n^2 - 1$ équilibres de Nash.

Proposition 7 *Il y a $\frac{(k+1)^n k(k+2)n(n-1)}{4}$ matrices de dimension n et de valeur dans [0,k] ayant $n^2 - 2$ équilibres de Nash.*

Preuve
Soit U une matrice de gain de dimension n, de valeur dans [0,k] ayant $n^2 - 2$ équilibres de Nash. Ceci veut dire que toutes les valeurs de U correspondent à des équilibres de Nash, sauf deux.

Ces deux valeurs non équilibres n'ont que deux possibilités de placement : soit elles sont toutes les deux sur la diagonale, soit elles sont symétriques toutes les deux (et donc en dehors de la diagonale).

Dans le cas où les non équilibres sont sur la diagonale, construisons la matrice. Pour ce qui concerne la colonne contenant sur sa diagonale la première valeur non équilibre, il y a $\frac{(k+1)k}{2}$ possibilités d'après la preuve de la propriété précédente. Pour la colonne contenant sur sa diagonale la seconde valeur non équilibre, il y a également $\frac{(k+1)k}{2}$ possibilités. Pour le reste de la matrice, il y a $(k+1)^{n-2}$ possibilités comme nous l'avons vu dans les propriétés précédentes. Il reste à compter les positions possibles des deux colonnes contenant les valeurs non équilibres. Le fait que ces valeurs soient sur la diagonale les rend totalement indépendantes l'une de

l'autre : le symétrique d'une valeur sur la diagonale est la valeur elle-même, et le fait qu'elle soit un équilibre ou non ne dépend que de sa propre colonne. Choisir les positions des deux colonnes revient aux arrangements de deux valeurs parmi n, soit C_n^2. Dans le cas où les deux valeurs sont sur la diagonale, on a donc $\frac{(k+1)k}{2} \times \frac{(k+1)k}{2} \times (k+1)^{n-2} \times C_n^2 = \frac{(k+1)^n k^2 n(n-1)}{8}$ matrices possibles.

Dans le cas où les deux valeurs non équilibres sont symétriques, elles sont dépendantes l'une de l'autre pour leur caractéristique d'équilibre de Nash. Il y a, à nouveau, deux cas possibles : soit une seule des deux valeurs est différente du maximum de sa colonne, soit les deux sont différentes.

Observons le cas où une seule des deux valeurs est différente du maximum de sa diagonale. Cette valeur peut se trouver n'importe où dans la matrice sauf sur la diagonale, soit $n(n-1)$ positions possibles. Pour la colonne contenant cette valeur, il y a $\frac{(k+1)k}{2}$ possibilités et $(k+1)^{n-1}$ pour le reste de la matrice. Au total, on a $n(n-1) \times \frac{(k+1)k}{2} \times (k+1)^{n-1} = \frac{(k+1)^n kn(n-1)}{2}$ matrices supplémentaires.

Dans le cas où les deux valeurs sont différentes du maximum de leur colonne, les positions possibles sont celles d'un couple de valeurs symétriques, soit une demi-matrice moins la diagonale, $\frac{n(n-1)}{2}$ positions. Pour la colonne contenant la première valeur différente du maximum de sa colonne, il y a $\frac{(k+1)k}{2}$ possibilités, idem pour la seconde colonne. Pour le reste de la matrice, il y a $(k+1)^{n-2}$ possibilités. Au total, on a $\frac{n(n-1)}{2} \times \frac{(k+1)k}{2} \times \frac{(k+1)k}{2} \times (k+1)^{n-2} = \frac{(k+1)^n k^2 n(n-1)}{8}$ matrices supplémentaires.

Finalement, en additionnant les nombres trouvés dans les trois cas relevés, il y a $\frac{(k+1)^n k^2 n(n-1)}{8} + \frac{(k+1)^n kn(n-1)}{2} + \frac{(k+1)^n k^2 n(n-1)}{8} = \frac{(k+1)^n kn(n-1)(k+2)}{4}$ matrices.

Remarque 1 *Dans le cas des matrices de dimension 2 (voir figure B.1), nous avons remarqué, dans tous les tests effectués, que la somme du nombre de matrices ayant 1 équilibre de Nash et de celui des matrices en ayant 3 est égal au nombre de matrices ayant 2 équilibres de Nash.*

Bibliographie

[1] Bruno Beaufils. *Modèles et simulations informatiques des problèmes de coopération entre agents.* Thèse pour le Doctorat de l'Université des Sciences et Technologies de Lille, 2000.

[2] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[3] R. Duncan Luce and Howard Raiffa. *Games Decisions : Introduction and Critical survey.* Dover Publications, 2 edition, 1985.

[4] William Poundstone. *Prisoner's Dilemma.* Doubleday, 1992.

[5] Robert Axelrod. *The evolution of cooperation.* Basic Books, New-York, USA, 1984.

[6] Jean-Louis Boursin. *Initiation à la théorie des jeux.* Montchrestien, Paris, 1998.

[7] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison-Wesley Publishing Company Inc., 1995, 1998.