

Table des matières

Introduction	1
I Existence et complexité des équilibres dans le jeu de placement	5
1 Introduction à la théorie des jeux	7
1.1 Les différents contextes de jeux	8
1.2 Formalisation du concept de jeu	9
1.2.1 Jeux sous forme normale	11
1.2.2 Jeux sous forme extensive ou arbres de jeux	11
1.2.3 Équilibres de Nash purs et mixtes	12
1.3 Les jeux de potentiel et de congestion	14
1.3.1 Jeux de potentiel	15
1.3.2 Jeux de congestion	16
1.3.3 Un cas particulier de jeux de congestion : l'ordonnancement	17
1.4 Calcul d'équilibres	20
1.4.1 Classes de complexité pour les jeux de congestion	20
1.5 Conclusion	22
2 Jeu de placement	23
2.1 Le contexte général du jeu de placement	24
2.1.1 Analogie avec l'ordonnancement classique	26
2.1.2 Équilibres dans le contexte atomique et non atomique	26
2.2 Cas d'étude	29
2.2.1 Jeu de placement sans restrictions (ou asymétrique)	29
2.2.2 Jeu de placement symétrique ($d_{i,j} = d_{j,i}$)	34
2.2.3 Jeu de placement symétrique non pondéré ($d_{i,j} = 1$)	35
2.3 Index des notations et conclusion	36
3 Équilibres du jeu de placement	39
3.1 Existence d'équilibres de Nash	40
3.1.1 Existence d'équilibres dans le jeu de placement asymétrique	40
3.1.2 Équilibres de Nash dans le jeu de placement symétrique	49
3.2 Calcul des équilibres de Nash	54
3.3 Structure des équilibres de Nash	59
3.3.1 Équilibres dans des cas particuliers de jeu de placement symétrique	60
3.3.2 Équilibres dans le jeu symétrique non pondéré atomique	65
3.4 Conclusion	72

II	Techniques d'apprentissage appliquées à des jeux de congestion et de potentiel	75
4	Techniques d'apprentissage d'équilibres	77
4.1	Dynamique de meilleure réponse	79
4.2	Dynamique de réplication	80
4.2.1	Notations et algorithme <i>LRI</i>	81
4.2.2	Propriétés de l'algorithme	82
4.3	Minimisation de regret et problème de bandits manchots . . .	83
4.3.1	Regrets et minimisation	83
4.3.2	Problème de Multi-Armed Bandit (ou Bandits Manchots)	85
4.3.3	Algorithme de minimisation de regret : <i>UCB</i>	87
4.4	Conclusion	89
5	Max-Cut et coordination d'interférences inter cellules	91
5.1	Coupe maximale dans un graphe non pondéré	93
5.1.1	Recherche d'une coupe maximale locale	97
5.1.2	Coupe localement maximale dans un graphe complet	98
5.1.3	Coupe maximum locale dans des graphes généraux .	101
5.1.4	Autres topologies de graphes et simulations	104
5.1.5	Lien avec l'interférence inter cellule	106
5.2	Coordination d'interférences inter cellules	107
5.2.1	Modélisation du système	108
5.2.2	Apprentissage distribué des équilibres de Nash purs . .	112
5.2.3	Convergence vers un équilibre de Nash pur de l'algorithme	113
5.2.4	Expérimentations	120
5.3	Conclusion	122
6	Minimisation de regret pour prédire la probabilité de violation de la qualité de service	123
6.1	Modèle de détection des échecs	124
6.1.1	Scénario de la négociation	126
6.1.2	Les échecs de SLA	129
6.2	Algorithme de sélection de l'offre du client	130
6.2.1	Adaptation du modèle de détection d'échec de SLA au MAB	131
6.2.2	Adaptation de la dynamique de réplication	134
6.3	Algorithme de sélection d'offres de QoS des NSPs	135
6.4	Résultats de simulations	136
6.4.1	Modèle des simulations	136
6.4.2	Résultats	138
6.5	Conclusion	143
	Conclusion	145

III	Annexe	149
7	Réallocation dynamique de tâches	151
7.1	Réallocation de tâches pondérées	152
7.2	Réallocation de tâches non pondérées	153
	Table des figures	157
	Liste des tableaux	159
	Bibliographie	161

Introduction

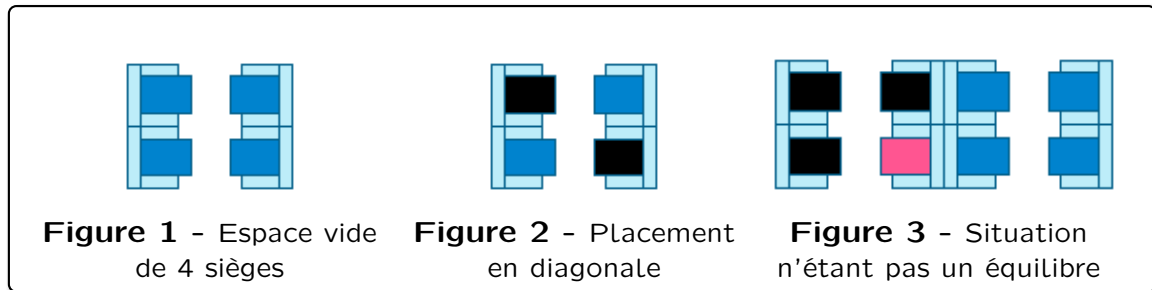
La théorie des jeux naît en tant que discipline au milieu du XX^{me} siècle. On attribue souvent celle-ci à la publication, en 1944, de l'ouvrage "*Theory of Games and Economic Behavior*" du mathématicien John Von Neumann et de l'économiste Oskar Morgenstern [Neumann et Morgenstern, 1944], mais une des premières publications dans le domaine date de 1838 [Cournot, 1838]. Dans cet ouvrage, intitulé "*Recherches sur les principes mathématiques de la théorie des richesses*", Antoine-Augustin Cournot aborde, dans son analyse du duopole (l'étude d'un marché opposant deux entreprises), un concept de solution qui est une version restreinte de *l'équilibre de Nash* tel qu'il sera formulé par John Forbes Nash dans les années 1950 [Nash, 1950, Nash, 1951].

En 1945, l'US Airforce crée la *RAND Corporation* (acronyme de "Research and Development") afin de comprendre les enjeux stratégiques du contexte militaire de l'époque, c'est-à-dire la guerre froide et les conflits nucléaires. Ce groupe a, dès le départ, réuni des chercheurs de toutes disciplines et rassemblé un grand nombre de personnalités scientifiques, notamment John Von Neumann, John Forbes Nash ou Lloyd Shapley. Les développements de la théorie des jeux au cours de la période 1945-1958 ont été tout à fait considérables, les enjeux n'étant rien de moins que d'éviter une escalade nucléaire conduisant à la destruction de la planète. L'équilibre de la terreur (ou destruction mutuelle assurée), élaborée durant cette époque de guerre froide, affirme que l'utilisation à grande échelle de l'arme nucléaire par l'Union soviétique ou Les États-Unis provoquerait à coup sûr la destruction des deux camps. En effet, le premier à lancer l'attaque afin de détruire l'autre est en quelque sorte assuré d'être détruit à son tour, annulant complètement l'intérêt d'une telle attaque.

La théorie des jeux est un outil destiné à la compréhension de systèmes dans lesquels des individus agissent dans leur propre intérêt. Un jeu est une situation où ces individus doivent faire un choix parmi un certain nombre d'actions possibles, le tout en fonction de règles du jeu définies à l'avance. Le résultat de ces choix, effectués par les différents individus, mène à l'issue du jeu à laquelle est associée une récompense pour chacun des participants. Dans de tels systèmes, le résultat d'une action ne dépend pas seulement de la décision de l'individu l'ayant effectuée, mais également de celles prises par les autres. La théorie des jeux permet d'analyser le comportement d'un individu qui doit prendre une décision, en tenant compte des décisions que peuvent prendre les autres individus de manière plus ou moins indépendante. Par exemple, pour un automobiliste devant décider entre deux chemins pour se rendre de son domicile au travail, ce choix de chemin dépendra des décisions prises par les autres automobilistes empruntant le même itinéraire.

Prenons un exemple de la vie courante, qui illustre le *jeu de placement* que nous étudierons par la suite : un individu souhaitant trouver la meilleure place dans un train. Considérons un wagon du train dont chaque compartiment est

formé par 4 sièges deux à deux face-à-face, comme représenté par la figure 1 ci-dessous. De manière logique, cet individu préférera s'installer seul dans l'un des espaces vides de 4 places plutôt que dans un espace déjà occupé.



Chacune des places disponibles dans le train correspond aux choix de placement que notre individu peut effectuer. Selon la proximité des autres personnes présentes dans le train, une place sera plus attrayante qu'une autre pour cet individu. L'individu en question a donc des préférences parmi les différents choix de places disponibles. Plus un choix lui est préférentiel, plus son bien-être sera important.

Malheureusement, en heure de pointe, il est rare de trouver un emplacement vide. L'individu doit alors revoir ses exigences à la baisse, mais préférera tout de même ne pas se placer directement à côté d'une autre personne. Il aura alors tendance à se placer comme sur la figure 2, où les sièges occupés sont indiqués en noir.

Il existe des configurations dans lesquelles notre individu peut se sentir mal à l'aise et de ce fait, souhaiter se déplacer afin d'augmenter son bien-être. Cette configuration est illustrée par la figure 3, où l'individu en rose préférera se déplacer vers le compartiment vide. Lorsqu'aucune des personnes à bord du train ne souhaite se déplacer, un état d'équilibre est atteint. En théorie des jeux, on appelle cet état *équilibre de Nash*. Néanmoins, cet équilibre est instable, l'arrivée, le départ ou le changement de place d'une personne peut rompre celui-ci.

Pourtant créée pour un cadre de stratégie économique et militaire, la théorie des jeux s'est révélée capitale dans l'analyse de plusieurs domaines scientifiques tels que l'économie, les sciences sociales, les sciences politiques, la biologie évolutive, mais également l'informatique.

Dans le cadre des réseaux informatiques du monde moderne où les systèmes sont de plus en plus grands, la mémoire partagée, impliquant une nécessité d'autonomie des différentes entités et agir dans un cadre distribué, la théorie des jeux a trouvé sa place. Les modèles très généraux utilisés en théorie des jeux peuvent être utilisés pour étudier un large éventail de phénomènes, par exemple les embouteillages sur l'itinéraire maison/travail ou l'allocation de processus sur un ensemble de processeurs. Nous tentons d'apporter des éléments de réponse dans des jeux tels que le jeu de placement dans un train, l'objectif étant de comprendre

- La notion d'équilibre pour le jeu donné et s'il en existe (chapitre 3) ;
- La complexité de calcul et les caractéristiques de ces équilibres (chapitres 5 et 6) ;
- s'il existe des algorithmes permettant la construction de ces équilibres de manière distribuée.

Organisation du document. Ce document se découpe en deux parties.

Dans la première partie, nous avons formalisé un type de jeu basé sur l'exemple de répartition dans un train que nous venons d'exposer. Ce jeu, que nous avons appelé *jeu de placement*, est une généralisation d'un jeu d'ordonnancement dans un graphe, où le placement du joueur ainsi que son voisinage influent sur son bien-être. Nous présentons les différents cas d'étude de notre jeu et considérons l'existence d'équilibres pour ces différentes variantes.

Nous introduirons dans le chapitre 1 les concepts de théorie des jeux évoqués précédemment. Nous définirons également une classe particulière de jeux, les jeux de potentiels, autour desquels s'articule ce document.

Le chapitre 2 se focalisera sur la formalisation du *jeu de placement*. Nous introduirons dans ce chapitre plusieurs cas d'étude du jeu de placement, correspondants à des changements topologiques du graphe associé au jeu. Parallèlement, nous illustrerons ces cas d'étude par de petits exemples permettant d'exhiber différentes propriétés de ces jeux.

Dans le chapitre, 3 nous discuterons l'existence et le calcul des équilibres des cas d'étude du jeu de placement. Nous nous sommes posé la question fondamentale étant de savoir s'il existait des équilibres de Nash et si tel est le cas s'ils étaient facilement calculables. Nous montrerons que, pour la version générale du jeu de placement, déterminer l'existence d'équilibre est NP-Complet. Néanmoins, en faisant une hypothèse restrictive sur ce jeu, nous obtiendrons un jeu de potentiel, que nous appellerons *jeu de placement symétrique*, dans lequel l'existence d'équilibres de Nash est garantie. Prouver l'existence d'équilibres dans un jeu ne nous garantit pas que ceux-ci soient facilement calculables. Nous nous intéresserons à la forme des équilibres de ce jeu réduit, ainsi qu'à la complexité liée à leur calcul, et prouverons que dans ce jeu trouver un équilibre est PLS-complet. Nous prouverons par la suite quelques résultats sur la structure des équilibres, qui nous mettront sur la piste du meilleur équilibre dans le *jeu de placement symétrique non pondéré*, correspondant au stable maximum du graphe associé au jeu.

Dans la deuxième partie de ce document, nous nous sommes intéressés à d'autres jeux de potentiel et au calcul distribué de leurs équilibres. Pour cela, nous utilisons des algorithmes, appelés algorithmes d'apprentissage, qui réalisent de petits ajustements (stochastiques ou déterministes) à chaque répétition du jeu.

Le chapitre 4 présentera la notion d'apprentissage. Une première dynamique, pour ces algorithmes d'apprentissage, qui est la plus naturelle correspond

à la dynamique de la meilleure réponse. Par la suite, nous introduirons une autre dynamique, appelée dynamique de réplcation, ainsi qu'un algorithme de minimisation de regret associé aux problèmes de *Multi-Armed Bandit*.

Dans le chapitre, 5 nous étudierons la convergence vers des équilibres dans un jeu de placement basé sur un problème issu de la théorie des graphes : Max-Cut. Le problème Max-Cut consiste à diviser en deux parties l'ensemble des sommets d'un graphe donné, de façon à maximiser la somme des poids des arêtes traversant la partition. Nous interpréterons ces coupes comme les équilibres de Nash purs d'un jeu de potentiel afin de calculer la convergence vers ceux-ci. Pour cela, nous développons un algorithme distribué probabiliste pour le calcul d'équilibres et montrons qu'en appliquant celui-ci sur un graphe complet de n sommets, une coupe localement maximale est atteinte en $O(\log \log n)$ étapes. En ce qui concerne un graphe quelconque, nous montrerons qu'une coupe localement maximale est atteinte en moyenne en $4\Delta|E|$ étapes, avec Δ le degré maximum du graphe.

Nous étendrons nos résultats à une application pour la gestion d'interférences dans des réseaux mobiles.

Parallèlement à l'étude du problème Max-Cut, nous nous sommes intéressés aux techniques de minimisation de regret. Néanmoins, celles-ci n'étaient pas applicables à Max-Cut étant donnée la taille réduite de l'ensemble des stratégies possibles. Nous nous focalisons alors, dans le chapitre 6, sur un autre système en concurrence : la prédiction de violation de qualité de service dans un problème de sélection d'offres de qualité de service parmi divers fournisseurs d'accès, où nous comparerons algorithmes d'apprentissage et minimisation de regret. Nous présentons un modèle pour le problème rencontré par un client au moment de choisir une offre de qualité de service pouvant rencontrer des échecs, dans un contexte où la fiabilité des fournisseurs peut changer. Ce problème est équivalent à celui d'un joueur ayant une utilité variable face à des machines dynamiques dans un problème de Multi-Armed Bandit. Nous proposerons une adaptation d'un algorithme de Multi-Armed Bandit et de la dynamique de réplcation permettant d'apprendre la fiabilité des fournisseurs d'accès.

Partie I

Existence et complexité des équilibres dans le jeu de placement



Résumé

Dans cette première partie, nous nous intéressons à la formalisation d'un jeu, basé sur le problème de placement dans le train présenté en introduction, et au calcul de ses équilibres.

Nous introduisons dans le chapitre 1 les concepts de théorie des jeux, tels que l'équilibre de Nash dont nous avons évoqué le principe en introduction. Nous définissons également des classes particulières de jeux, les jeux de potentiel et de congestion, autour desquels

s'articule ce document.

Dans le chapitre 2, nous formalisons un jeu de potentiel, que nous avons appelé *jeu de placement*, inspiré de l'exemple de répartition dans un train. Nous introduisons plusieurs cas d'étude du jeu de placement, que nous illustrons par de petits exemples permettant d'exhiber différentes propriétés de ces jeux.

Enfin, nous discutons de l'existence et du calcul des équilibres de ces cas d'étude dans le chapitre 3.

Introduction à la théorie des jeux

La théorie algorithmique des jeux [Nisan et al., 2007] est intéressée par la compréhension de l'ensemble du système lorsque celui-ci peut converger vers des situations rationnelles, appelées équilibres de Nash. L'équilibre de Nash, introduit en 1950 [Nash, 1950], est une notion centrale de solution pour les jeux stratégiques. La théorie algorithmique des jeux s'intéresse à la manière d'obtenir ce type d'équilibres ainsi qu'à la complexité de leur calcul. Cette théorie s'est principalement concentrée sur la caractérisation des équilibres dans des classes particulières de jeux [Rosenthal, 1973], la qualité des équilibres [Koutsoupias et Papadimitriou, 1999], certains algorithmes distribués convergeant vers des équilibres [Berenbrink et al., 2012].

Dans ce chapitre, nous présentons quelques jeux classiques, issus de la théorie des jeux, tout en introduisant ses concepts de base. Dans une première partie, nous formaliserons les différents termes abordés ci-dessus (jeu, joueur, stratégies, utilités, équilibre de Nash). Puis nous nous concentrerons sur une classe de jeux particulière : les jeux de potentiel, autour desquels va s'articuler ce document. Nous aborderons par la suite la complexité algorithmique des équilibres et leur calcul.

Sommaire

2.1	Le contexte général du jeu de placement	24
2.1.1	Analogie avec l'ordonnancement classique	26
2.1.2	Équilibres dans le contexte atomique et non atomique	26
2.2	Cas d'étude	29
2.2.1	Jeu de placement sans restrictions (ou asymétrique)	29
2.2.2	Jeu de placement symétrique ($d_{i,j} = d_{j,i}$)	34
2.2.3	Jeu de placement symétrique non pondéré ($d_{i,j} = 1$)	35
2.3	Index des notations et conclusion	36

1.1

Les différents contextes de jeux

La théorie des jeux permet d'analyser l'interaction d'entités rationnelles, appelées agents ou joueurs (un être humain, un animal, une entreprise, etc.), poursuivant des buts qui leur sont propres. Dans une situation d'interaction entre différents joueurs, nous pouvons dégager les propriétés suivantes, qui donneront lieu à des modèles différents :

- la relation existante entre les joueurs (possibilité d'établir des coalitions, *jeu coopératif ou non coopératif*) ;
- le déroulement temporel du jeu (*séquentiel ou simultané*) ;
- les informations dont disposent les joueurs (*information complète ou incomplète, parfaite ou imparfaite*).

Les relations entre les joueurs.

Il existe des jeux dits, *jeux coopératifs*, où l'on s'intéresse aux coalitions (aussi appelées alliances) que les joueurs peuvent former pour coordonner leurs actions dans un but commun. Le problème de ce type de jeux réside dans la répartition de la récompense du jeu au sein de la coalition [Shapley, 1953] et la tendance des joueurs à vouloir la quitter. En effet, il est rare que la coordination des actions des joueurs implique de meilleures récompenses pour l'ensemble des joueurs de la coalition. Les jeux coopératifs impliquent au contraire une part de sacrifice de l'intérêt propre d'un ou plusieurs joueurs au profit d'un bien-être commun, jugé supérieur.

Nous ne développerons pas ce type de jeux, ce document se concentrant sur des *jeux non coopératifs*.

Par opposition aux jeux coopératifs, les jeux où aucune alliance n'est possible sont appelés *jeux non coopératifs*. Dans le cas d'un jeu non coopératif, les joueurs sont en concurrence les uns avec les autres, comme c'est le cas pour des jeux à deux joueurs tels que les échecs, pierre papier ciseau, etc., ou des jeux à plusieurs joueurs comme le poker ou un automobiliste dont le temps de trajet dépend du trafic routier, c'est-à-dire des autres automobilistes/joueurs. Le bien-être d'un joueur dépend alors des choix de ses concurrents. Attention jeu non coopératif ne veut pas dire que les joueurs ont des buts antagonistes, par exemple les automobilistes n'ont pas pour but de nuire aux autres, en revanche des joueurs de poker ou d'échec ont tout intérêt à conduire leur adversaire à la défaite. De plus, il ne faut pas automatiquement associer un jeu où les communications entre les joueurs sont possibles à un jeu coopératif, la communication n'implique pas forcément la possibilité de créer des coalitions.

Le déroulement du jeu.

Notons dans un premier temps qu'un jeu peut être joué une seule ou plusieurs fois. Dans le premier cas il s'agit d'un jeu dit *statique*, dans le second nous parlons de *jeu répété*. Pour un jeu répété, nous avons alors un même jeu (même nombre de joueurs, mêmes ensembles d'actions, mêmes fonctions de récompense) répété à chaque étape.

Le jeu peut se dérouler de manière simultanée ou séquentielle. Dans un jeu simultané, aussi appelé *jeu stratégique*, les joueurs choisissent leur action à effectuer en même temps que chacun des autres joueurs une fois pour toutes au début du jeu (ou à chaque début de tour de jeu dans le cas d'un jeu répété) et perçoivent les résultats du jeu en même temps. Au contraire, si un jeu implique des choix successifs de la part des joueurs nous sommes dans le cas d'un *jeu séquentiel*.

Les informations connues des joueurs.

Quel que soit le type d'environnement auquel un joueur fait face, le jeu puisse être

- à *information complète* : le joueur connaît les règles du jeu, et tout le déroulement du jeu jusqu'à sa prise de décision ;
- à *information incomplète (ou partielle)* : un ou plusieurs de ces éléments est inconnu du joueur.

De plus, un jeu est dit à *information parfaite* lorsque chaque joueur a une connaissance de tout l'historique du jeu, des actions passées des autres joueurs et des récompenses associées.

1.2**Formalisation du concept de jeu**

Nous nous concentrons ici sur des *jeux statiques* finis, aussi appelés jeux non répétés. En théorie algorithmique des jeux, la plupart des concepts concernent des jeux finis (le nombre de joueurs et l'ensemble de stratégies sont finis). Néanmoins, beaucoup de ces concepts peuvent être étendus et généralisés à des jeux dits continus, où l'ensemble des stratégies des joueurs porte sur un espace continu.

Les joueurs.

Un jeu comprend un ensemble d'*individus rationnels* (appelés également joueurs ou agents) qui peuvent prendre des décisions, ils interagissent et influent sur les résultats des autres. Les joueurs sont généralement supposés

- économiquement rationnels : c'est-à-dire qu'ils agissent de manière égoïste afin de tenter d'arriver à la situation qui leur est la plus avantageuse ;
- raisonnant de manière stratégique : ils prennent en compte leurs connaissances du jeu et les anticipations du comportement des autres preneurs de décisions.

Nous notons $N = \{1, \dots, n\}$ l'ensemble des joueurs.

Les stratégies.

Nous distinguons ici deux types de stratégies, les stratégies pures et mixtes. À chaque tour de jeu, les joueurs doivent décider d'une action à effectuer parmi un ensemble d'actions possibles. Un joueur i appartenant à N dispose donc d'un ensemble fini d'actions possibles S_i . Ce sont ces actions que nous appelons par la suite stratégies. Le vecteur des stratégies choisies pour chaque joueur est appelé *profil de stratégies, ou configuration*.

Stratégies pures : nous appelons *stratégies pures* les actions possibles des joueurs. Nous notons $S = S_1 \times \dots \times S_n$ le profil de stratégies pures.

Stratégies mixtes : une *stratégie mixte* $q_i = (q_{i,1}, q_{i,2}, \dots, q_{i,|S_i|})$ du joueur i correspond à une distribution de probabilités sur les stratégies pures. L'action $k \in S_i$ est choisie avec une probabilité $q_{i,k} \in [0, 1]$, avec $\sum_{k=1}^{|S_i|} q_{i,k} = 1$. Notons que toute stratégie pure k peut être considérée comme une stratégie mixte e_k , où e_k dénote le vecteur de probabilités avec la $k^{ième}$ composante égale à 1 et les autres à 0.

Nous notons $\Delta(S_i)$ l'ensemble des stratégies mixtes du joueur i , et de ce fait $\Delta(S) = \prod_{i=1}^n \Delta(S_i)$ l'espace de toutes les stratégies mixtes. Un profil de stratégies $q = (q_1, \dots, q_n) \in \Delta(S)$ précise les stratégies de tous les joueurs, q_i correspondant à la stratégie mixte du joueur i .

Les utilités.

À chaque profil de stratégies est associée une récompense (un gain ou une perte) pour chaque joueur, découlant d'une *fonction d'utilité* à valeur pour chaque combinaison possible de stratégies. La sélection d'une stratégie au détriment d'une autre implique un résultat, et donc une satisfaction du joueur, différente. La notion de bien-être ou de satisfaction d'un joueur dans une situation donnée est traduite par cette fonction d'utilité qui associe une série de choix préférentiels à des valeurs numériques. Pour chaque joueur i , la fonction d'utilité est notée u_i . Nous notons $u = \{u_1, \dots, u_n\}$ l'ensemble des fonctions d'utilité des joueurs. Comme les joueurs interagissent les uns avec les autres, leur propre utilité u_i est une fonction de toutes les autres stratégies des joueurs : $u_i : S \rightarrow \mathbb{R}$.

Utilités pour les profils de stratégies mixtes.

La fonction d'utilité u_i d'un joueur i peut être étendue à l'ensemble de ses stratégies mixtes $\Delta(S_i)$. Nous écrivons alors $u_i(q)$ comme étant l'espérance de u_i pour les stratégies mixtes de q , avec $u_i(q) = \sum_{s \in S} Pr(s|q) \cdot u_i(s)$.

1.2.1 Jeux sous forme normale

La *forme normale* d'un jeu est un modèle pour un jeu non coopératif simultané statique ou répété avec information presque complète (dû à la simultanéité du jeu). Dans le cas de jeu séquentiel, nous lui préférons la *forme extensive* que nous verrons par la suite.

Un jeu sous forme normale est entièrement spécifié par le triplet $G = (N, S, u)$, avec

- $N = \{1, \dots, n\}$ un ensemble de joueurs ;
- $S_i = \{s_1, \dots, s_{n_i}\}$ un ensemble de stratégies pures pour le joueur i ;
- $u_i : S_1 \times \dots \times S_n \mapsto \mathbb{R}$ une fonction d'utilité pour chaque joueur i , qui à chaque profil de stratégies associe l'utilité du joueur i .

Cette notation est appelée *jeu sous forme normale (ou forme stratégique)*.

Pour un jeu à deux joueurs sous forme normale et ayant un nombre fini et suffisamment restreint de stratégies, il est possible de donner la matrice des utilités du jeu.

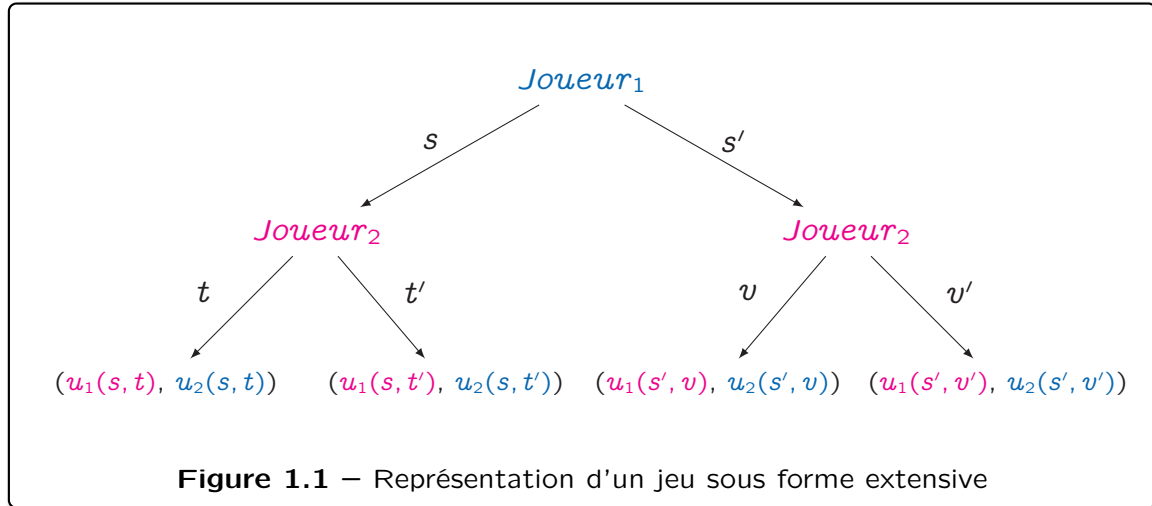
		Joueur ₁	
		s	s'
Joueur ₂	t	$(u_1(s, t), u_2(s, t))$	$(u_1(s', t), u_2(s', t))$
	t'	$(u_1(s, t'), u_2(s, t'))$	$(u_1(s', t'), u_2(s', t'))$

Tableau 1.1 – Représentation d'un jeu sous forme matricielle

Cette matrice se présente sous la forme d'un tableau à $|S_\ell|$ lignes et $|S_c|$ colonnes, où S_ℓ et S_c sont l'ensemble des stratégies à la disposition des joueurs ℓ et c , respectivement ligne et colonne. Chaque case du tableau est remplie par un vecteur, donnant l'utilité de chaque joueur pour chaque profil de stratégie, qui correspond à la paire de stratégies de la ligne et colonne de la case considérée.

1.2.2 Jeux sous forme extensive ou arbres de jeux

La *forme extensive* d'un jeu permet de modéliser un jeu non coopératif ayant un déroulement séquentiel et à information complète et parfaite. La représentation d'un jeu sous forme extensive se fait au moyen d'un arbre, c'est-à-dire un graphe orienté sans circuit. Cet arbre est composé d'un ensemble de sommets, représentant un joueur ou un instant de décision, connecté à d'autres sommets par des arcs symbolisant les actions possibles.



Dans la figure 1.1, deux joueurs $Joueur_1$ et $Joueur_2$ jouent l'un à la suite de l'autre avec deux stratégies possibles s ou s' pour $Joueur_1$ et t et t' pour $Joueur_2$. Le jeu se termine par la récupération de la récompense lorsque l'on arrive sur une feuille de l'arbre. Les flèches situées au bas de l'arbre représentent les quatre issues du jeu possibles équivalant aux vecteurs présents dans la forme matricielle d'un jeu.

1.2.3 Équilibres de Nash purs et mixtes

Les définitions qui suivent sont données dans le cas de jeux sous forme normale, mais peuvent être étendues aux jeux sous forme extensive. Nous reprenons pour cela les notations $G = (N, S, u)$ de la section 1.2.1.

Dans un jeu, chaque joueur i a pour objectif de minimiser son coût c_i ou bien de maximiser son utilité u_i . À partir de maintenant, nous allons nous concentrer sur les utilités et supposer que chaque joueur i cherche à maximiser sa propre utilité u_i , les définitions seront données en ce sens.



Notation de théorie des jeux

Étant donné un profil de stratégie $s \in S$ et un joueur $i \in N$, nous noterons, de façon abusive, s_{-i} le vecteur des stratégies jouées par tous les joueurs hormis le joueur i , soit $s_{-i} = \{s_1, s_2, \dots, s_{i-1}, s_{i+1}, \dots, s_n\}$. Par extension, nous noterons $s' = (s'_i, s_{-i})$ le vecteur, issu de s , où seule la stratégie du joueur i a été modifiée et remplacée par s'_i , soit $s' = \{s_1, s_2, \dots, s_{i-1}, s'_i, s_{i+1}, \dots, s_n\}$.

Afin de définir l'équilibre de Nash, dont nous avons discuté en introduction, nous introduisons tout d'abord la notion de *bonne réponse*. Une bonne réponse pour un joueur est une situation où ce joueur a intérêt à changer de stratégie afin d'améliorer son utilité, les stratégies des autres joueurs étant fixées (voir la définition 1.1). Nous pouvons également introduire la notion

de *meilleure réponse* pour un joueur, celle-ci correspondant à l'action (ou ensemble d'actions) offrant au joueur l'utilité optimale parmi les bonnes réponses possibles. Une meilleure réponse étant également une bonne réponse, lorsqu'un joueur peut améliorer son utilité en changeant de stratégie, nous parlons de meilleure réponse par abus de notation. Lorsqu'aucun joueur ne peut effectuer de meilleure réponse, le jeu se trouve dans un état appelé *équilibre de Nash* (voir la définition 1.3).



Définition 1.1 – Bonne réponse (stricte)

Étant donné un profil de stratégies s , un joueur i peut effectuer une *bonne réponse* par rapport à s , s'il existe une stratégie s'_i telle que

$$u_i(s'_i, s_{-i}) > u_i(s).$$



Définition 1.2 – Meilleure réponse (stricte)

Étant donné un profil de stratégies s , un joueur i peut effectuer une *meilleure réponse* par rapport à s , s'il existe un ensemble S'_i de stratégies tel que

$$\forall s'_i \in S'_i, u_i(s'_i, s_{-i}) = \max_{s \in S'_i} u_i(s).$$

Un équilibre de Nash pur (NE) est un profil de stratégies pures dans lequel aucun joueur ne peut, de manière unilatérale, choisir une action différente lui permettant d'améliorer son utilité. À l'équilibre, aucun joueur n'a intérêt à dévier unilatéralement de sa stratégie, étant données les stratégies jouées par les autres joueurs.



Définition 1.3 – Équilibre de Nash

Un *équilibre de Nash pur* est un profil de stratégies pures pour lequel aucun joueur ne peut effectuer de meilleure réponse.

Nous pouvons également introduire une notion plus forte d'équilibre qu'est l' ε -équilibre de Nash.



Définition 1.4 – ε -équilibre de Nash

Un ε -*équilibre de Nash* est un état dans lequel aucun joueur ne peut, en changeant sa stratégie, améliorer son utilité par un facteur multiplicatif $1 + \varepsilon$, étant données les stratégies jouées par les autres joueurs.

L'équilibre de Nash a été introduit dans le but de prédire les états vers lesquels converge un jeu si les joueurs agissent de manière rationnelle. Si un état stationnaire tel que l'équilibre de Nash est atteint, alors d'après la définition du

principe de meilleure réponse aucun joueur n'a intérêt à modifier sa stratégie. En effet, dans le cas contraire, le joueur ayant modifié sa stratégie voit son utilité se dégrader, il n'agirait alors pas de manière rationnelle.

Néanmoins, il existe des jeux ne possédant pas toujours un équilibre de Nash pur, comme par exemple pierre/papier/ciseaux. Nash a cependant prouvé que dans tout jeu fini il existe un équilibre de Nash en stratégies mixtes [Nash, 1950].



Définition 1.5 – Équilibre de Nash mixte

Un *équilibre de Nash mixte* est un profil de stratégies mixtes $Q = (q_1, \dots, q_n) \in \Delta(S)$ tel que

$$\forall q'_i \in \Delta(S_i), u_i(q_i, Q_{-i}) \geq u_i(q'_i, Q_{-i})$$

avec $\Delta(S_i)$ l'ensemble des stratégies du joueur i .

Il existe différents types de jeux, du classique jeu symétrique à deux joueurs en passant par les jeux à somme nulle, les jeux à n joueurs, etc.

Dans un jeu à somme constante, les différents joueurs se partagent une utilité fixe, comme une certaine somme d'argent, etc. Les jeux à somme nulle [Neumann et Morgenstern, 1944] sont un cas particulier de ces jeux à somme constante pour lesquels, comme leur nom l'indique, la somme des utilités des joueurs vaut 0. Ce sont principalement des jeux à deux joueurs dans lesquels les intérêts de chacun sont parfaitement antagonistes. Cette classe de jeux correspond à des jeux compétitifs tels que les jeux de société ou jeux de cartes par exemple les échecs ou le poker.

Dans ce qui suit, nous nous focalisons sur une classe particulière et importante de jeux, appelés jeux de potentiel. Ces jeux possèdent plusieurs propriétés intéressantes, notamment le fait que les équilibres de Nash purs existent toujours et qu'il existe des algorithmes itératifs qui convergent vers ceux-ci (voir la dynamique de meilleure réponse que nous présentons au chapitre 4).

1.3

Les jeux de potentiel et de congestion

Dans ce document, nous nous sommes concentrés principalement sur une classe de jeux non coopératifs simultanés, appelés *jeux de congestion*, présentée par Rosenthal en 1973 [Rosenthal, 1973]. Rosenthal décrit ces jeux de congestion comme "une classe de jeux possédant toujours au moins un équilibre de Nash", mais le nom de jeu de congestion fut donné plus tard par Monderer et Shapley

en 1996 [Monderer et Shapley, 1996b] lorsqu'ils présentèrent les jeux de potentiel, dont les jeux de congestion sont un cas particulier.

1.3.1 Jeux de potentiel

Un jeu est dit être un *jeu de potentiel* si l'incitation de tous les joueurs à modifier leur stratégie peut être exprimée en utilisant une fonction globale unique, appelée *fonction de potentiel*. Il existe plusieurs sous-classes de jeux de potentiel, les *jeux de potentiel ordinal*, *exact* et *pondéré*.

Cette *fonction de potentiel*, notée Φ , de l'ensemble des stratégies pures et à valeurs dans $\mathbb{R}$, est telle qu'étant donné un profil de stratégies pures, si un joueur change de stratégie alors la variation de son utilité implique une variation de même type pour Φ .

Dans le cas d'un *jeu de potentiel ordinal*, seul le signe de la variation est conservé, une variation positive (resp. négative) de l'utilité implique également une variation positive de la fonction de potentiel (resp. négative). Pour les *jeux de potentiel exact*, le signe et la valeur de la variation sont identiques. Quant aux *jeux de potentiel pondéré*, la valeur de la variation est proportionnelle à un certain poids.



Définition 1.6 – Jeu de potentiel ordinal

Un jeu $G = (N, S, u)$ est un *jeu de potentiel ordinal* si pour tout profil de stratégies pures s et s'_i une stratégie du joueur i différente de s_i , alors

$$\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) > 0 \text{ si et seulement si } u_i(s_i, s_{-i}) - u_i(s'_i, s_{-i}) > 0$$



Définition 1.7 – Jeu de potentiel exact

Un jeu $G = (N, S, u)$ est un *jeu de potentiel exact* si pour tout profil de stratégies pures s et s'_i une stratégie du joueur i différente de s_i , alors

$$\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) = u_i(s_i, s_{-i}) - u_i(s'_i, s_{-i})$$

Un jeu de potentiel exact est donc également un jeu de potentiel ordinal.



Définition 1.8 – Jeu de potentiel pondéré

Un jeu est un *jeu potentiel pondéré* s'il existe des poids $(w_i)_{i \in N}$, et Φ tels que pour toutes les stratégies pures s et s'_i une stratégie du joueur i différente de s_i , alors

$$\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) = w_i(u_i(s_i, s_{-i}) - u_i(s'_i, s_{-i}))$$

Par définition d'un jeu de potentiel, dans le cas d'une maximisation de l'utilité,

un profil $S = (s_1, \dots, s_n)$ est un *équilibre de Nash pur* si et seulement si $\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) \geq 0$, pour tout joueur i et pour toute autre stratégie pure s'_i de ce joueur. En particulier, un équilibre de Nash peut être vu comme un maximum local pour la fonction Φ . En effet, par définition d'un équilibre de Nash, aucun joueur ne peut changer de stratégie afin d'améliorer son utilité, Φ ne peut alors pas être augmentée.

Pour une minimisation d'un coût, un profil $S = (s_1, \dots, s_n)$ est un équilibre de Nash pur si et seulement si $\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) \leq 0$, pour tout joueur i et pour toute autre stratégie pure s'_i de ce joueur. En ce cas, de la même manière, l'équilibre de Nash peut être vu comme un minimum local de Φ .



Théorème 1.1 – [Monderer et Shapley, 1996b]

Tout jeu de potentiel ordinal possède un équilibre de Nash pur.

Preuve : Soit un jeu de potentiel ordinal $G = (N, S, u)$, Φ sa fonction de potentiel et où chaque joueur i cherche à maximiser son utilité u_i . L'idée de cette preuve est de montrer qu'il existe une suite finie de profils de stratégies où un unique joueur effectue une action et améliore son utilité entre chaque profil. Cette suite menant alors à un équilibre de Nash.

Prenons une suite de profils $S^0, S^1, S^2, \dots, S^k, \dots$ telle que pour chaque k un seul joueur i dispose d'une stratégie différente entre S^k et S^{k+1} et avec $u_i(S^k) < u_i(S^{k+1})$. Si un profil n'est pas un équilibre de Nash alors, cela implique qu'il existe un joueur i pouvant effectuer une meilleure réponse. Tant qu'une meilleure réponse peut être obtenue, nous pouvons construire une telle suite.

Par la définition d'un jeu de potentiel ordinal, $\Phi(S^0) < \Phi(S^1) < \dots < \Phi(S^k) < \dots$. Comme l'espace des stratégies est fini Φ est bornée, de ce fait, la suite de profils est finie et le dernier élément de celle-ci correspond à un équilibre de Nash. $\square$

1.3.2 Jeux de congestion

Les *jeux de congestion* sont un cas particulier de jeux de potentiels et ont été introduits par Rosenthal en 1973 [Rosenthal, 1973]. Les *jeux de congestion* sont une classe de jeux où les joueurs participants choisissent leurs ressources simultanément, les ressources pouvant être partagées entre plusieurs joueurs. L'utilité d'un joueur dépend de la ressource choisie, mais également du nombre

de joueurs ayant choisi cette même ressource. Le coût de chaque ressource dépend d'une fonction de congestion.



Définition 1.9 – Jeu de congestion [Rosenthal, 1973]

Un *jeu de congestion* est un jeu non coopératif défini par un 4-uplet $(N, R, (d_r)_{r \in R}, (S_i)_{i \in N})$ avec :

- $N = \{1, \dots, n\}$ l'ensemble des joueurs ;
- R l'ensemble des ressources ;
- pour chaque $r \in R$, $d_r : N \leftarrow \mathbb{Z}$ une fonction de délai ;
- pour chaque $i \in N$, S_i contient des sous-ensembles de R ;
- la fonction de coût c_i définie par $c_i(S) = \sum_{r \in S_i} d_r(X_r(S))$ où $X_r(S)$ est le nombre de joueurs dont la stratégie contient r sur le profil S .

Rosenthal [Rosenthal, 1973] a montré à l'aide d'une *fonction de potentiel exacte* que tout jeu de congestion possède au moins un équilibre de Nash, néanmoins cela n'induit pas une convergence rapide vers cet équilibre. Monderer et Shapley ont prouvé la réciproque en 1996 [Monderer et Shapley, 1996a].



Théorème 1.2 – [Monderer et Shapley, 1996b]

Les jeux de congestion sont des jeux de potentiel exact. Tout jeu de potentiel exact est équivalent à un jeu de congestion.

Dans ce qui suit, nous introduisons des jeux de congestion particuliers : les jeux d'ordonnancement. De par leur appartenance aux jeux de congestion, ceux-ci sont également des jeux de potentiel. C'est autour de ce cas particulier de jeux que s'organise le reste de cette partie du document.

1.3.3 Un cas particulier de jeux de congestion : l'ordonnancement

L'*ordonnancement* (aussi appelé équilibrage de charges ou "*load balancing*") est un problème d'allocation de tâches à un ensemble de ressources (ou machines) capables de les réaliser, l'objectif étant que cette allocation soit la plus équitable possible en termes de charges.

Prenons un exemple concret simple : considérons le cas où chaque ressource représente un canal de communication avec une bande passante limitée. Le problème est d'attribuer chaque demande de bande passante entrante vers l'un des canaux. L'affectation d'une demande à un certain canal de communication augmente la charge sur ce canal, soit le pourcentage de la bande passante utilisée. La charge est augmentée pendant la durée associée à la requête.

De base, l'ordonnancement n'est pas un jeu, mais nous le formulons sous forme de jeu afin de faire apparaître son caractère distribué. Dans ce qui suit, nous nous intéressons au problème de la réallocation dynamique de tâches.

Modèle du jeu d'ordonnancement.

Nous nous intéressons au problème de la réallocation dynamique de tâches de manière distribuée, c'est-à-dire un équilibrage de charges étant donné un placement initial aléatoire des différentes tâches sur les ressources. Nous disposons de n tâches, auxquelles est associé un poids $w_i \geq 1$ pour tout $i \in \{1, \dots, n\}$, et m ressources uniformes. Deux ressources sont uniformes si, étant donnée une tâche, celle-ci est traitée de manière identique sur les deux ressources, quelle que soit sa pondération.

Le modèle se base sur une approche considérant chaque tâche comme un joueur, soit un individu rationnel dont l'objectif est son intérêt propre. Chaque joueur cherche alors à minimiser son coût, qui est le poids total de la ressource sur laquelle il se trouve, appelé la charge. Le problème qui nous intéresse est donc un problème de minimisation. De ce fait, nous ne notons pas la fonction d'utilité u_i pour un joueur i , mais c_i afin de représenter le coût subi par ce joueur.

L'assignement des n tâches aux ressources est représenté par un vecteur de taille m , $X(t) = (X_1(t), \dots, X_m(t))$ où chaque composante $X_j(t)$ représente la charge de la ressource $j \in \{1, \dots, m\}$ à la fin de l'étape t , c'est-à-dire la somme des poids des tâches allouées à cette ressource.

Nous définissons le jeu sous forme normale suivant $G = (N, S, c)$, avec

- $N = \{1, \dots, n\}$ l'ensemble des tâches (les joueurs) ;
- $S = \{1, \dots, m\}$ les ressources (l'ensemble de stratégies pures, identiques pour tout joueur i) ;
- $c_i = \{X_j | s_i = j\}$ une fonction de coût pour chaque joueur i , qui à chaque profil de stratégies associe le coût du joueur i .

À chaque étape, les tâches ont pour but de se placer sur une ressource leur permettant de diminuer leur coût. Lorsqu'aucune ressource de ce type n'existe pour chacune des tâches, nous sommes dans un équilibre de Nash.



Propriété 1.1 – Équilibre de Nash pour le jeu d'ordonnancement

Un assignement est un équilibre de Nash pour la tâche i , pour tout $i \in N$, si

$$\forall j \in \{1, \dots, m\}, c_i \leq X_j + w_i.$$

La tâche i ne peut améliorer sa situation en migrant vers n'importe quelle autre ressource.

Nous pouvons également définir la notion d' ε -équilibre de Nash.


Propriété 1.2 – ε -équilibre de Nash pour le jeu d'ordonnancement

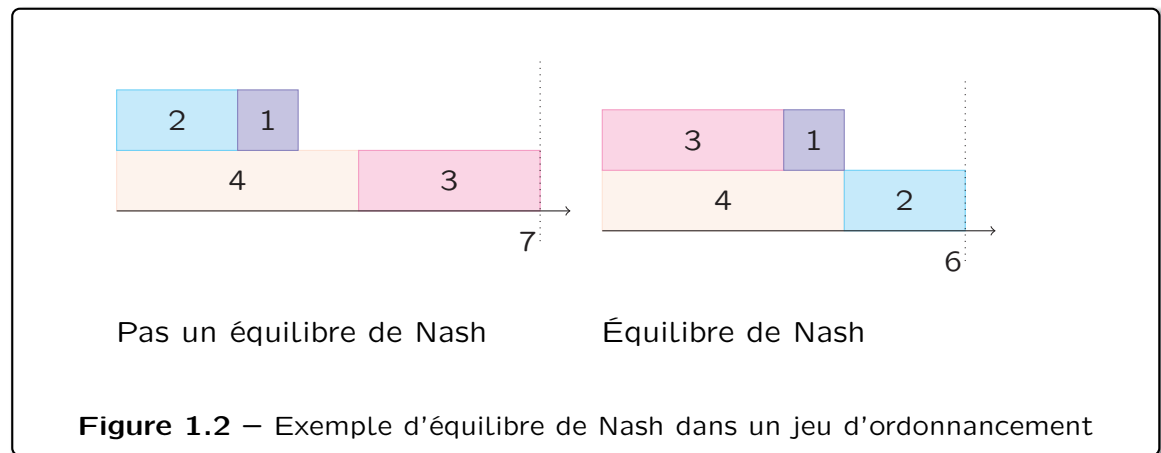
Pour tout $\varepsilon \in [0, 1]$, un assignement est un ε -équilibre de Nash pour la tâche i , pour tout $i \in N$, si

$$\forall j \in \{1, \dots, m\}, c_i \leq X_j + (1 + \varepsilon)w_i.$$

Exemple d'équilibre de Nash dans un jeu d'ordonnancement

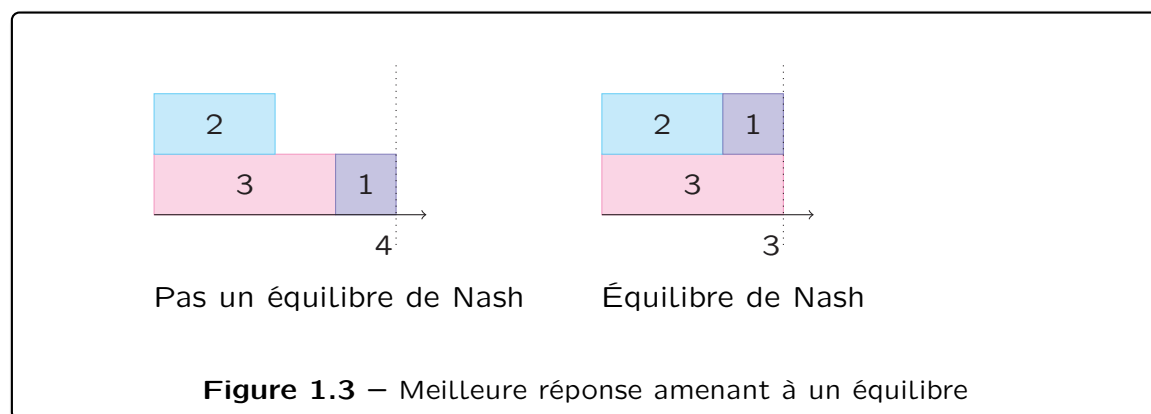
Prenons un jeu d'ordonnancement avec deux ressources et 4 tâches de poids 1, 2, 3 et 4 qui sont nos joueurs. Les représentations de la figure 1.2 montrent un état qui n'est pas un équilibre de Nash et un autre état correspondant à un tel équilibre. En effet dans la représentation de gauche le joueur 3 ressent un coût $c_3 = 4 + 3 = 7$, le fait de changer de stratégie pour la seconde ressource lui permettrait de baisser son coût à $c_2 = 2 + 1 + 3 = 6$. Comme au moins un joueur peut effectuer une meilleure réponse, cette configuration n'est pas un équilibre.

Au contraire, en prenant la configuration de droite, nous pouvons remarquer qu'aucun mouvement d'un des joueurs ne permet d'améliorer le coût de celui-ci. Cet état est donc un équilibre de Nash.



Prenons maintenant un jeu d'ordonnancement avec deux ressources et 3 tâches de poids 1, 2 et 3. Les représentations de la figure 1.3 montrent un état qui n'est pas un équilibre de Nash à gauche et un autre état, issu du premier qui correspond à un équilibre de Nash.

Il existe donc une meilleure réponse permettant de passer de la configuration initiale à un équilibre de Nash. Nous pouvons montrer que le joueur 1 peut effectuer une meilleure réponse, et ainsi améliorer son utilité, en se déplaçant vers la deuxième ressource.



Soit $S = \{s_1, s_2, s_3\}$ le profil de stratégies correspondant à la figure de gauche. Soit s_1 la stratégie actuelle du joueur et s'_1 la stratégie consistant à choisir l'autre ressource, alors $c_1(s_1, S_{-1}) - c_1(s'_1, S_{-1}) = 4 - 3 = 1$. Comme $c_1(s_1, S_{-1}) > c_1(s'_1, S_{-1})$ en changeant de stratégie pour la stratégie s'_1 le joueur 1 effectue une meilleure réponse.

De plus, quel que soit le joueur il n'existe pas de changement de stratégie lui permettant de diminuer son coût depuis cette nouvelle configuration. Cet état, correspondant à la figure de droite, est donc un équilibre de Nash.

Nous avons présenté une classe de jeux pour laquelle des équilibres de Nash existent, néanmoins nous n'avons pas abordé le calcul d'un tel équilibre. Dans ce qui suit, nous discutons du calcul d'équilibres de Nash et de leur complexité dans les jeux de congestion.

1.4

Calcul d'équilibres

L'équilibre d'un jeu est l'état du système que l'on cherche atteindre, il est alors naturel d'en étudier la complexité. L'étude des équilibres de Nash a permis d'introduire de nouvelles classes de complexité, telles que *PLS* ("*Polynomial Local Search*") [Johnson et al., 1988] et *PPAD* ("*Polynomial Parity Arguments on Directed graphs*") [Papadimitriou, 1994, Daskalakis et al., 2006].

Dans certains jeux, la recherche d'équilibres de Nash est un problème complet pour ces classes, et donc algorithmiquement difficile.

1.4.1 Classes de complexité pour les jeux de congestion

Les problèmes de recherche d'optimum local sont dans la classe *PLS*. En particulier, trouver un équilibre de Nash dans les jeux de congestions appartient

à la classe PLS, l'équilibre correspondant à un trouver un optimum local de la fonction de Rosenthal [Rosenthal, 1973].



Définition 1.10 – Problème d'optimisation locale

Un *problème d'optimisation locale* Π consiste en un ensemble d'instances $\mathcal{I}$, un ensemble de solutions satisfiables $\mathcal{F}(I)$ pour chaque instance I dans $\mathcal{I}$, et une fonction objectif $f : \mathcal{F}(I) \mapsto \mathbb{Z}$.

Chaque solution $s \in \mathcal{F}(I)$ possède un voisinage $\mathcal{N}(s, I) \subseteq \mathcal{F}(I)$. Pour une instance $I \in \mathcal{I}$, le problème est de trouver une solution $s \in \mathcal{F}(I)$ telle que toute solution $s' \in \mathcal{N}(s, I)$ n'a pas une valeur $f(s')$ plus grande pour le problème de maximisation (plus petite pour le problème de minimisation).

Un problème d'optimisation locale Π est dans la classe *PLS*, définie dans [Johnson *et al.*, 1988], s'il existe trois algorithmes polynomiaux A , B , et C tels que, pour toute instance I dans $\mathcal{I}$,

- l'algorithme A trouve en temps polynomial une solution faisable dans $\mathcal{F}(I)$;
- pour une solution s , l'algorithme B vérifie si s est faisable ($s \in \mathcal{F}(I)$) et si c'est le cas, calcule sa valeur $f(s)$ en temps polynomial ;
- pour une solution faisable $s \in \mathcal{F}(I)$, l'algorithme C calcule en temps polynomial une solution $s' \in \mathcal{N}(s, I)$ fournissant une meilleure valeur pour f ($f(s') > f(s)$ pour un problème de maximisation ou $f(s') < f(s)$ pour une minimisation) s'il en existe une, ou informe que s est localement optimale dans le cas contraire.

Un problème $\Pi \in PLS$ est *PLS-réductible* à un problème $\Pi' \in PLS$ s'il existe deux fonctions σ et Ψ calculables en temps polynomial telles que

- la fonction σ associe une instance I de Π à une instance I' de Π' ;
- la fonction Ψ associe à une instance I' de Π' un couple (s, I) , où s est une solution faisable de I de Π , tel que pour toutes les instances I de Π , et pour tous les optimums locaux s^* de $\sigma(I)$, la solution $\Phi(s^*, I)$ est un optimum local pour I .

De plus, un problème $\Pi \in PLS$ est *PLS-complet* si chaque problème dans *PLS* est *PLS-réductible* à Π .

Concernant le calcul d'équilibre de Nash dans un jeu de congestion, nous pouvons citer le théorème suivant :



Théorème 1.3 – [Fabrikant *et al.*, 2004]

Trouver un équilibre de Nash pur dans un jeu de congestion est PLS-complet.

1.5

Conclusion

Nous avons vu dans ce chapitre les notions de base de la théorie des jeux. Nous nous sommes intéressés plus particulièrement à un type de jeux, que sont les jeux de congestion, et introduit la complexité du calcul d'un équilibre de Nash dans de tels jeux. Dans la suite de cette partie, nous définissons un jeu, que nous nommerons *jeu de placement*, qui est une généralisation d'un jeu d'ordonnancement dans un graphe. Notre objectif est de calculer des équilibres et déterminer leur complexité dans le jeu de placement. Le chapitre suivant définit les différentes variantes du jeu, tandis que le chapitre 3 abordera le calcul des équilibres.

2

Jeu de placement

Dans le chapitre précédent, nous avons discuté d'une classe de jeux : Les jeux de congestion, et en particulier Les jeux d'ordonnancement.

Nous introduisons dans ce chapitre une généralisation d'un jeu d'ordonnancement dans un graphe, où le placement du joueur ainsi que la pondération des arêtes voisines influent sur son coût. Pour ce jeu, que nous appellerons par la suite jeu de placement, nous nous sommes posé la question fondamentale qui est de savoir s'il existait des équilibres de Nash et si tel est le cas s'ils étaient facilement calculables. Dans le cas contraire, nous chercherons la classe de complexité à laquelle notre jeu appartient.

Ce chapitre a pour vocation de définir le jeu de placement et ses différentes variantes. Par la suite, nous répondrons aux questions évoquées ci-dessus.

Sommaire

3.1	Existence d'équilibres de Nash	40
3.1.1	Existence d'équilibres dans le jeu de placement asymétrique	40
3.1.2	Équilibres de Nash dans le jeu de placement symétrique	49
3.2	Calcul des équilibres de Nash	54
3.3	Structure des équilibres de Nash	59
3.3.1	Équilibres dans des cas particuliers de jeu de placement symétrique	60
3.3.2	Équilibres dans le jeu symétrique non pondéré atomique	65
3.4	Conclusion	72

Dans l'introduction, nous avons illustré un problème de la vie courante : où un individu aura tendance à se placer dans un train de manière à ne pas être incommodé par les autres personnes présentes ? Ce problème peut se modéliser sous la forme d'un jeu d'ordonnancement dans lequel le voisinage d'une ressource a un impact sur la charge de celle-ci. En effet, le bien-être d'un individu occupant un siège dans un train dépendra de ses préférences (sens de la marche, fenêtre, radiateur, etc.), mais sera également directement impacté par la présence d'autres individus sur les sièges voisins. Dans cet exemple, le siège correspond à la ressource occupée par l'individu et son bien-être s'apparente à la charge ressentie. C'est en s'intéressant à ce problème que nous avons introduit le jeu de placement que nous définissons dans ce chapitre.

Dans leurs travaux [Even-Dar *et al.*, 2007], les auteurs étudient le nombre d'étapes nécessaires pour atteindre un équilibre de Nash pur dans un scénario d'équilibrage de charge pour une variété de modèles d'ordonnancement. Les auteurs ont montré que ces différentes variantes sont des jeux de potentiel. Un jeu d'ordonnancement où les joueurs ne peuvent se déplacer que dans leur voisinage a également été étudié dans [Diekmann *et al.*, 1999]. À notre connaissance, il n'existe pas de variante des jeux d'ordonnancement où le coût d'une ressource dépend de son voisinage. Cependant, la modélisation du jeu de placement est semblable à une classe de jeu, les "Graphical games" [Kearns *et al.*, 2001]. Cette forme graphique est une représentation d'un jeu sur un graphe non-orienté, où chaque joueur est un sommet et a une utilité dépendant de lui-même ainsi que de ses voisins dans le graphe. D'autres jeux stratégiques, tels que les jeux de coordination sur des graphes [Apt *et al.*, 2014], s'intéressent à l'impact du voisinage, mais dans un contexte de coordination des joueurs. Ces jeux se placent dans un graphe non orienté où les sommets correspondent aux joueurs du jeu. Chaque joueur dispose, pour stratégies, d'un ensemble de couleurs parmi lesquelles choisir, son utilité dépendant du nombre de ses voisins ayant choisi la même stratégie. Dans [Apt *et al.*, 2014], les auteurs ont montré que ces jeux de coordination sur des graphes sont des jeux de potentiel exact.

Dans la section suivante, nous définissons le jeu de placement dans son cadre le plus général. Par la suite, nous présenterons différents cas d'étude que nous analyserons dans le chapitre 3.

2.1

Le contexte général du jeu de placement

Le *jeu de placement*, dans son cadre le plus général, considère un graphe orienté dont les arcs sont pondérés et un ensemble de joueurs qui doivent se

répartir sur celui-ci. L'objectif d'un joueur est de se placer sur un sommet du graphe, sachant que le voisinage de l'emplacement sélectionné a un impact sur son coût et qu'il cherche à minimiser celui-ci.

Chaque joueur ℓ doit choisir une stratégie s_ℓ parmi l'ensemble des stratégies que représente V , l'ensemble des sommets du graphe. Une stratégie s_ℓ du joueur ℓ correspond alors au choix d'un sommet $v \in V$. Notons que, par la suite, nous considérons chaque joueur ℓ correspondant à une tâche dont le poids est noté p_ℓ . La charge x_v d'un sommet v correspond alors à la somme des poids des joueurs ayant choisi v comme stratégie, soit

$$x_v(S) = \sum_{\ell \in N, s_\ell = v} p_\ell.$$

Étant donné le graphe $G = (V, A)$ et $N = \{1, \dots, n\}$ l'ensemble des joueursⁱ, chaque sommet $v \in V$ dispose d'une fonction de coût $C_v(x_v)$. Le coût $C_v(x_v)$ dépend de x_v la charge du sommet v , auquel s'additionne le coût de chaque sommet voisin pondéré par les arcs. En effet, à chaque arc partant d'un sommet i vers j est attribuée une pondération $d_{i,j}$ correspondant à l'impact de la charge du sommet i sur le coût de j . Nous avons donc $C^\ell(S)$ le coût du joueur ℓ pour un profil de stratégie S qui est égal au coût du sommet choisi et vaut :

$$C^\ell(S) = C_{s_\ell}(S) = \sum_{j \in N^-[i]} d_{j,i} x_j(S),$$

où $N^-[i]$ est le voisinage intérieur fermé du sommet i , c.-à-d., $N^-[i] = \{j : (j, i) \in A\}$, et sachant $s_\ell = i$.



Définition 2.1 – Définition du jeu de placement

- Joueurs : $N = \{1, \dots, n\}$ les joueurs ;
- Stratégies : $S = V = \{1, \dots, m\}$ l'ensemble des sommets.
Notons $S = \{s_1, \dots, s_n\}$ un profil de stratégies et $x(S) = (x_1(S), \dots, x_m(S))$ où $x_v(S)$ est la charge du sommet v selon le profil de stratégies S ;
- Coût : $\forall \ell \in N, C^\ell(S) = C_{s_\ell}(S) = \sum_{j \in N^-[i]} d_{j,i} x_j(S)$, sachant $s_\ell = i$.

L'objectif de chaque joueur ℓ de N est de minimiser son coût $C^\ell(S) = C_{s_\ell}(S)$.

Nous noterons D la matrice des coefficients des arcs. Ainsi, nous définirons notre jeu par G le graphe associé, D la matrice de coefficients et n le nombre de joueurs. Cela nous donne pour notation d'un jeu de placement (G, D, n) .

i. Contrairement aux notations usuelles de théorie des graphes, ici, la notation n désigne le nombre de joueurs, et non le nombre de sommets qui est m .

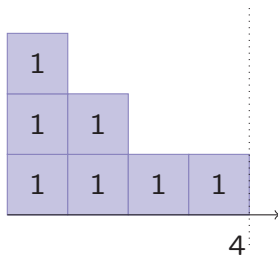
2.1.1 Analogie avec l'ordonnancement classique

Avec ces notations, le jeu d'ordonnancement classique correspond à un graphe sans arcs, excepté les boucles (donc, pour tout sommet j , $\Gamma_j = \{j\}$ est le voisinage ouvert du sommet j), et des fonctions de coût de la forme suivante pour chaque sommet j du graphe :

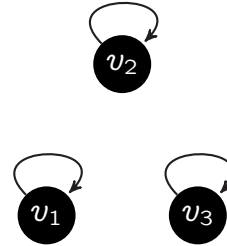
$$C_j(x_1, \dots, x_n) = \frac{x_j}{v_j}$$

où $v_j > 0$ est la *vitesse* associée au sommet (à la "ressource") j , avec $d_{j,j} = \frac{1}{v_j}$. Pour tout couple de sommets i et j tel que $i \neq j$ nous avons $d_{i,j} = d_{j,i} = 0$.

Nous prenons ici un exemple d'ordonnancement où la vitesse associée à chaque ressource vaut 1 et où les tâches sont de poids unitaire et identiques. La figure 2.1 contient un exemple d'ordonnancement de 7 joueurs sur trois ressources à gauche, que nous avons formulé comme un jeu de placement à droite.



Ordonnancement sur trois ressources, la première ayant une charge de 1, la deuxième deux tâches soit une charge totale de 2 et une troisième ressource ayant quatre tâches dont le total des charges vaut 4.



Ordonnancement sous forme de jeu de placement avec trois sommets formant un stable où v_1 correspond à la première ressource, v_2 la deuxième et v_3 la troisième avec de ce fait $x_{v_1} = 1$, $x_{v_2} = 2$ et $x_{v_3} = 4$.

Figure 2.1 — Exemple d'ordonnancement et son équivalent sous forme de jeu de placement

2.1.2 Équilibres dans le contexte atomique et non atomique

Nous distinguons deux contextes, le contexte *atomique* et le contexte *non atomique*, dans lesquels les vecteurs de charges sont calculés différemment. Le jeu atomique peut être considéré comme un jeu fini discret tandis que le jeu non atomique représente une "dédiscretisation" du jeu atomique, soit un jeu fini, mais continu.

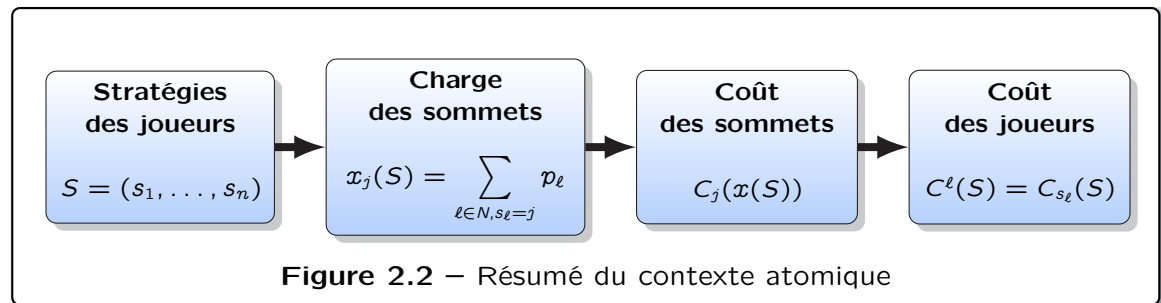
2.1.2.1 Jeu atomique

Dans le contexte atomique, un ensemble $N = \{1, \dots, n\}$ de joueurs (ou tâches) est donné, chaque joueur $\ell \in N$ disposant d'un poids p_ℓ . Pour un joueur, une *stratégie* est le choix d'un sommet du graphe. L'ensemble des stratégies est donc $V = \{1, 2, \dots, m\}$. Un profil de stratégies S est un choix d'une stratégie par joueur :

$$S = (s_1, s_2, \dots, s_n) \in V^n.$$

À chaque profil de stratégies S est associé :

- la charge de toute ressource $j \in V$, par $x_j(S) = \sum_{\ell \in N, s_\ell = j} p_\ell$;
- le profil de charges $x(S) = (x_1(S), x_2(S), \dots, x_m(S))$;
- le coût associé à chaque ressource $j : C_j(x(S))$, où C_j est la fonction de coût associée au sommet j .



Dans le cas d'un jeu atomique, l'équilibre correspond à un équilibre de Nash comme défini dans le chapitre 1.

2.1.2.2 Jeu non atomique

Dans le cadre non atomique maintenant, nous regardons directement les distributions de charges $(x_j)_{j \in V}$ où les x_j sont des réels positifs ou nuls et la somme de toutes les charges est égale à une constante fixée (que nous supposons égale à 1). Les coûts sont alors définis directement par les fonctions de coûts.

Si la recherche d'équilibre dans un jeu atomique s'apparente à trouver un équilibre de Nash, cette notion n'est plus valable dans le cas d'un jeu non

atomique.



Définition 2.2 – Équilibre non atomique

Un *équilibre non atomique* dans un graphe d'ordonnancement $(V, E, (C_j)_{j \in V})$ est une répartition de charges $x_j \geq 0$, où la charge totale vérifie

$$\sum_{j \in V} x_j = 1,$$

telle que pour tout couple (j, j') de sommets nous ayons

$$C_j(x_1, x_2, \dots, x_m) = C_{j'}(x_1, x_2, \dots, x_m)$$

avec $x_j > 0$ et $x_{j'} > 0$.

Comme évoqué précédemment, un jeu non atomique peut être vu comme la version continue d'un même jeu atomique. Nous pouvons alors faire une analogie avec les jeux de congestion et les jeux de Wardrop [Correa et Stier-Moses, 2011] qui sont une version continue de ces derniers et possèdent leur propre notion d'équilibre.

Nous pouvons également définir la notion d'équilibre fort non atomique, qui peut être apparenté à la notion d' ε -équilibre de Nash pour le jeu non atomique.



Définition 2.3 – Équilibre fort non atomique

Un *équilibre fort non atomique* dans un graphe d'ordonnancement $(V, E, (C_j)_{j \in V})$ est une répartition de charges $x_j \geq 0$, où la charge totale vérifie

$$\sum_{j \in V} x_j = 1,$$

telle que pour tout $\varepsilon > 0$ et pour tout couple (j, j') de sommets nous ayons

$$x_j - \varepsilon > 0 \Rightarrow C_j(x_1, \dots, x_m) \leq C_{j'}(x_1, x_2, \dots, x_j - \varepsilon, \dots, x_{j'} + \varepsilon, \dots, x_m),$$

autrement dit, aucune quantité ε de charges ne peut diminuer son coût en se déplaçant du sommet j au sommet j' .

Nous avons défini le cadre général du jeu de placement et évoqué les questions de recherche d'équilibres. Pour cela, nous avons étudié plusieurs variantes de notre jeu de placement que nous définissons dans la section suivante.

2.2

Cas d'étude

Dans un premier temps, nous nous sommes concentrés sur une version du jeu de placement qui, comme dans le cadre général, considère une pondération des arcs correspondant à l'impact du voisinage sur un sommet. Nous avons considéré ce type de jeu dans deux types de graphes, les graphes orientés et non orientés que nous nommons par la suite respectivement *jeu de placement asymétrique* et *jeu de placement symétrique*. Enfin, par extension du jeu symétrique, nous nous sommes placés dans le cadre d'un graphe non orienté et sans pondération que nous appelons *jeu de placement symétrique non pondéré*.

**Remarques**

Sauf formellement explicité du contraire, dans les preuves des jeux de placement :

- nous considérons que le nombre de joueurs est supérieur au nombre de sommets du graphe ($n > m$) ;
- pour chaque sommet i , l'impact de la charge du sommet sur lui-même vaut 1, c.-à-d., $d_{i,i} = 1$;
- dans la représentation graphique, nous ne faisons pas apparaître les boucles sur chacun des sommets ;
- pour des simplifications de calculs, nous considérons que pour un joueur ℓ , le poids du joueur vaut :

$$p_\ell = \begin{cases} 1 & \text{si le jeu est atomique} \\ \frac{1}{n} & \text{si le jeu est non atomique.} \end{cases}$$

2.2.1 Jeu de placement sans restrictions (ou asymétrique)

Comme nous en avons discuté dans le chapitre 1, contrairement aux jeux de congestion et autres jeux de potentiel certains jeux ne possèdent pas toujours d'équilibre de Nash. C'est le cas pour le jeu de placement sans restriction, aussi appelé jeu de placement asymétrique. Nous allons analyser deux exemples de jeu de placement asymétrique, un premier exemple possédant un équilibre de Nash et un second pour lequel nous montrerons qu'un tel équilibre n'existe pas.

Ce premier exemple, que nous nommerons *exemple 1*, est un jeu à n joueurs composé de deux sommets v_1 et v_2 . Chacun de ces sommets a un arc sortant vers l'autre sommet de coefficient $k > 1$, c'est-à-dire $d_{v_1, v_2} = d_{v_2, v_1} = k$. Nous avons également une boucle en v_1 et en v_2 dont le poids vaut 1, soit $d_{v_1, v_1} = d_{v_2, v_2} = 1$. Chaque joueur ℓ , a pour poids $p_\ell = 1$. Cet exemple est illustré par la figure 2.3. Par souci de clarté, nous ne représenterons pas les boucles de poids unitaire sur les figures.

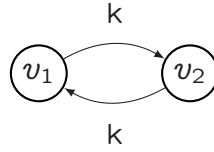


Figure 2.3 – Représentation du jeu dans l'exemple 1



Lemme 2.1 – Équilibres pour le jeu exemple 1

Le jeu à n joueurs exemple 1, dans sa version atomique, admet deux équilibres de Nash purs consistant à placer les n joueurs sur le même sommet.

Preuve : Preuve par l'absurde.

Supposons qu'il existe une configuration S_1 qui soit un équilibre tel que v_1 et v_2 sont tous deux chargés. Soient $x_{v_1}(S_1)$ et $x_{v_2}(S_1)$ leur charge dans cette configuration S_1 , alors

$$\begin{cases} C_{v_1}(S_1) = x_{v_1}(S_1) + k \cdot x_{v_2}(S_1) \\ C_{v_2}(S_1) = x_{v_2}(S_1) + k \cdot x_{v_1}(S_1) \end{cases}$$

Soit S'_1 la configuration obtenue à partir de S_1 lorsqu'une tâche ℓ passe de v_1 à v_2 . En S_1 la tâche ℓ avait pour coût $C^\ell(S_1) = C_{v_1}(S_1)$. Comme $x_{v_1}(S'_1) = x_{v_1}(S_1) - 1$ et $x_{v_2}(S'_1) = x_{v_2}(S_1) + 1$, cela implique que

$$\begin{aligned} C^\ell(S'_1) &= C_{v_2}(S'_1) \\ &= x_{v_2}(S'_1) + k \cdot x_{v_1}(S'_1) \\ &= (x_{v_2}(S_1) + 1) + k \cdot (x_{v_1}(S_1) - 1) \\ &= x_{v_2}(S_1) + k \cdot x_{v_1}(S_1) + 1 - k \\ &< C^\ell(S_1) \text{ (car } k > 1) \end{aligned}$$

S_1 n'est donc pas un équilibre : tant que les deux sommets restent chargés, il est plus intéressant pour chaque tâche de migrer vers le sommet voisin.

Les deux sommets ne peuvent pas être chargés en même temps, mais un stable est-il un équilibre ?

Supposons maintenant, sans perte de généralité, S_2 une configuration telle que $x_{v_1}(S_2) = n$ et $x_{v_2}(S_2) = 0$. Soit S'_2 la configuration obtenue à partir de S_2 lorsqu'une tâche ℓ passe de v_1 à v_2 . En S_2 la tâche ℓ avait pour coût $C^\ell(S_2) = C_{v_1}(S_2) = x_{v_1}(S_2) + k \cdot x_{v_2}(S_2) = n$. Comme $x_{v_1}(S'_2) = x_{v_1}(S_2) - 1 = n - 1$ et $x_{v_2}(S'_2) = x_{v_2}(S_2) + 1 = 1$ cela implique que

$$\begin{aligned} C^\ell(S'_2) &= C_{v_2}(S'_2) \\ &= x_{v_2}(S'_2) + k \cdot x_{v_1}(S'_2) \\ &= (x_{v_2}(S_2) + 1) + k \cdot (x_{v_1}(S_2) - 1) \\ &= 1 + k \cdot (n - 1) \\ &> C^\ell(S_2) \end{aligned}$$

Le sommet v_2 n'est pas attractif, aucune tâche n'a intérêt à se déplacer vers le sommet vide. Le profil de stratégies S_2 est donc un équilibre.

Étant donné que les deux sommets ne peuvent être chargés en même temps, les équilibres possibles sont des stables, dans lesquels chaque joueur ne ressent que la charge du sommet occupé. □

Pour l'instance explicitée dans l'exemple 1, un équilibre de Nash existe. Nous allons maintenant considérer un exemple de jeu similaire, mais ne possédant pas d'équilibre de Nash.

Soit le jeu de placement (G, D, n) composé d'un cycle orienté de 3 sommets v_1, v_2 et v_3 où tous les coefficients des arcs sont égaux à un entier $k > 1$, tel que représenté sur la figure 2.4. Nous avons alors pour chaque arc $d_{v_1, v_2} = d_{v_2, v_3} = d_{v_3, v_1} = k$. Nous avons également une boucle en v_1, v_2 et en v_3 dont le poids vaut 1, soit $d_{v_1, v_1} = d_{v_2, v_2} = d_{v_3, v_3} = 1$. Le jeu considère n joueurs, chaque joueur ℓ ayant pour poids $p_\ell = 1$. Tout comme dans l'exemple précédent, les boucles de poids unitaire ne sont pas représentées sur la figure par souci de clarté. Ce jeu, que nous nommerons exemple 2, ne dispose pas d'équilibre de Nash.

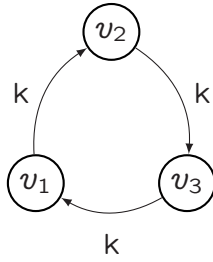


Figure 2.4 – Représentation du jeu dans l'exemple 2



Lemme 2.2 – Équilibres pour le jeu exemple 2

Le jeu à n joueurs exemple 2, dans sa version atomique n'admet pas d'équilibre de Nash pur.

Preuve : Étant donné le graphe de la figure 2.4, il existe trois types de configurations possibles :

- **Cas 1** : un seul des sommets est chargé ;
- **Cas 2** : deux des sommets sont chargés ;
- **Cas 3** : tous les sommets sont chargés.

Pour un état S , étant donnés $x_{v_1}(S)$, $x_{v_2}(S)$ et $x_{v_3}(S)$ les charges des sommets v_1 , v_2 et v_3 les coûts de chacun sont les suivants :

$$\begin{cases} C_{v_1}(S) = x_{v_1}(S) + k \cdot x_{v_3}(S) \\ C_{v_2}(S) = x_{v_2}(S) + k \cdot x_{v_1}(S) \\ C_{v_3}(S) = x_{v_3}(S) + k \cdot x_{v_2}(S) \end{cases}$$

Raisonnons par l'absurde en supposant chacune de ces configurations comme étant un équilibre de Nash pur, une configuration dépendant du nombre de sommets chargés.

Cas 1. Sans perte de généralité, supposons une configuration S_1 qui soit un équilibre de Nash, et telle que $x_{v_1}(S_1) = n$ et $x_{v_2}(S_1) = x_{v_3}(S_1) = 0$. Soit S'_1 la configuration obtenue à partir de S_1 lorsqu'une tâche ℓ passe de v_1 à v_3 . En S_1 la tâche ℓ avait pour coût $C^\ell(S_1) = C_{v_1}(S_1) = x_{v_1}(S_1) + k \cdot x_{v_3}(S_1) = n$.

Les différences de charges de S_1 à S'_1 sont les suivantes

$$\begin{cases} x_{v_1}(S'_1) = x_{v_1}(S_1) - 1 = n - 1 \\ x_{v_2}(S'_1) = x_{v_2}(S_1) = 0 \\ x_{v_3}(S'_1) = x_{v_3}(S_1) + 1 = 1 \end{cases}$$

De cela est déduit le coût de la tâche ℓ après migration

$$\begin{aligned} C^\ell(S'_1) &= C_{v_3}(S'_1) \\ &= x_{v_3}(S'_1) + k \cdot x_{v_2}(S'_1) \\ &= (x_{v_3}(S_1) + 1) + k \cdot x_{v_2}(S_1) \\ &= 1 \\ &< C^\ell(S_1) \end{aligned}$$

La configuration S_1 n'est donc pas un équilibre. Il ne peut y avoir un seul sommet chargé dans le graphe.

Cas 2. Sans perte de généralité, supposons qu'il existe une configuration S_2 qui soit un équilibre tel que v_1 et v_2 sont tous deux chargés et v_3 ait une charge nulle. Soit S'_2 la configuration obtenue à partir de S_2 lorsqu'une tâche ℓ passe de v_1 à v_2 . Dans l'état S_2 , la tâche ℓ avait pour coût $C^\ell(S_2) = C_{v_1}(S_2)$. Comme $x_{v_1}(S'_2) = x_{v_1}(S_2) - 1$, $x_{v_2}(S'_2) = x_{v_2}(S_2) + 1$ et $x_{v_3}(S'_2) = x_{v_3}(S_2) = 0$ cela implique que

$$\begin{aligned} C^\ell(S'_2) &= C_{v_2}(S'_2) \\ &= x_{v_2}(S'_2) + k \cdot x_{v_1}(S'_2) \\ &= (x_{v_2}(S_2) + 1) + k \cdot (x_{v_1}(S_2) - 1) \\ &= x_{v_2}(S_2) + k \cdot x_{v_1}(S_2) + 1 - k \\ &< C^\ell(S_2) \text{ (car } k > 1) \end{aligned}$$

S_2 n'est donc pas un équilibre. Il ne peut y avoir uniquement deux sommets chargés dans le graphe.

Cas 3. Supposons maintenant une configuration S_3 qui soit un équilibre tel que v_1 , v_2 et v_3 sont chargés. Sans perte de généralité, soit S'_3 la configuration obtenue à partir de S_3 lorsqu'une tâche ℓ passe de v_1 à v_2 . Dans l'état S_3 , la tâche ℓ avait pour coût $C^\ell(S_3) = C_{v_1}(S_3)$. Comme $x_{v_1}(S'_3) = x_{v_1}(S_3) - 1$, $x_{v_2}(S'_3) = x_{v_2}(S_3) + 1$ et $x_{v_3}(S'_3) = x_{v_3}(S_3)$ cela implique que

$$\begin{aligned} C^\ell(S'_3) &= C_{v_2}(S'_3) \\ &= x_{v_2}(S'_3) + k \cdot x_{v_1}(S'_3) \\ &= (x_{v_2}(S_3) + 1) + k \cdot (x_{v_1}(S_3) - 1) \\ &= x_{v_2}(S_3) + k \cdot x_{v_1}(S_3) + 1 - k \\ &< C^\ell(S_3) \text{ (car } k > 1) \end{aligned}$$

S_3 n'est donc pas un équilibre. Les trois sommets ne peuvent donc être simultanément chargés.

Nous avons montré, sans perte de généralité, qu'aucune des trois configurations possibles n'était un équilibre de Nash. Il n'existe donc pas d'équilibre pur dans le jeu illustré par la figure 2.4. $\square$

En faisant une hypothèse restrictive sur le jeu de placement asymétrique, nous définissons une variante de celui-ci, le jeu de placement symétrique.

2.2.2 Jeu de placement symétrique ($d_{i,j} = d_{j,i}$)

Le *jeu de placement symétrique* est un jeu de placement avec la particularité suivante : $\forall (i, j) \in V^2, d_{i,j} = d_{j,i}$. De ce fait entre chaque paire de sommets i et j il y a au plus une arête $e = (i, j)$ de poids $d_{i,j} = d_{j,i}$. Nous nous plaçons désormais dans un graphe non orienté, $G = (V, E)$.

Nous illustrons l'existence d'équilibre de Nash dans ce type de jeu par un exemple représenté sur la figure 2.5.

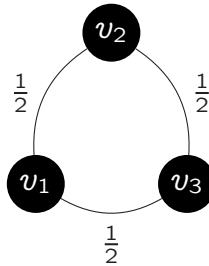


Figure 2.5 – Exemple d'équilibre de Nash dans un jeu de placement symétrique
 $x_{v_1} = \frac{n}{3}$, $x_{v_2} = \frac{n}{3}$ et $x_{v_3} = \frac{n}{3}$

Soit le jeu de placement (G, D, n) composé d'un cycle non orienté de 3 sommets v_1 , v_2 et v_3 où tous les coefficients des arêtes sont égaux à un entier $k = \frac{1}{2}$. Nous avons alors $\forall (v_i, v_j) \in E, d_{v_i, v_j} = d_{v_j, v_i} = k$. Nous avons également une boucle en v_1 , v_2 et en v_3 , non représentée sur la figure, dont le poids vaut 1, soit $d_{v_1, v_1} = d_{v_2, v_2} = d_{v_3, v_3} = 1$.



Lemme 2.3 – Équilibres dans un cycle de 3 sommets

Le jeu de placement à n joueurs dont le graphe est un cycle de 3 sommets admet, dans sa version atomique, un équilibre de Nash pur.

Preuve : Supposons un nombre de joueurs n étant un multiple de 3. Étant données les charges $x_{v_1} = x_{v_2} = x_{v_3} = \frac{n}{3}$ des sommets v_1, v_2 et v_3 , les coûts de chacun des sommets sont les suivants :

$$\begin{cases} C_{v_1} = x_{v_1} + \frac{1}{2} \cdot x_{v_2} + \frac{1}{2} \cdot x_{v_3} = \frac{2n}{3} \\ C_{v_2} = \frac{1}{2} \cdot x_{v_1} + x_{v_2} + \frac{1}{2} \cdot x_{v_3} = \frac{2n}{3} \\ C_{v_3} = \frac{1}{2} \cdot x_{v_1} + \frac{1}{2} \cdot x_{v_2} + x_{v_3} = \frac{2n}{3} \end{cases}$$

Dans le cas où n ne serait pas un multiple de 3, nous aurions chacune des charges x_{v_1}, x_{v_2} et x_{v_3} comprises entre $\frac{n}{3} - 1$ et $\frac{n}{3} + 1$. Nous pouvons montrer qu'il n'existe aucun changement de stratégie possible permettant d'améliorer le coût d'un des joueurs. Ces calculs simples étant répétitifs nous ne les détaillerons pas. De plus dans le chapitre 3 nous montrons que ce jeu est un jeu de potentiel et admet de ce fait des équilibres de Nash. $\square$

2.2.3 Jeu de placement symétrique non pondéré ($d_{i,j} = 1$)

La variante présentée ci-dessous découle du jeu de placement symétrique. Nous ajoutons la contrainte suivante : $\forall (i, j) \in E, d_{i,j} = 1$. Rappelons que $\forall i \in V, d_{i,i} = 1$ et que pour chaque joueur ℓ , $p_\ell = 1$.

Les figures 2.6 et 2.7 représentent un exemple d'équilibre de Nash dans un jeu de placement symétrique non pondéré. Les arêtes étant de poids 1 nous n'affichons pas la pondération sur celles-ci. Comme précédemment n est supposé multiple de 3 dans la figure 2.6 et pair dans la figure 2.7.

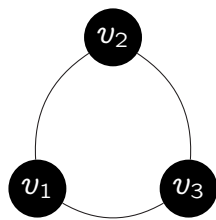


Figure 2.6 – Exemple d'équilibre de Nash dans un jeu de placement symétrique non pondéré
 $x_{v_1} = \frac{n}{3}, x_{v_2} = \frac{n}{3}$ et $x_{v_3} = \frac{n}{3}$

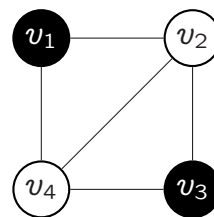


Figure 2.7 – Exemple d'équilibre de Nash dans un jeu de placement symétrique non pondéré
 $x_{v_1} = \frac{n}{2}$ et $x_{v_3} = \frac{n}{2}$

La figure 2.6 est un équilibre sur un graphe de trois sommets, où chaque sommet ressent sa charge ainsi que celle de chaque autre sommet du graphe. En répartissant la charge équitablement entre les trois sommets, avec $x_{v_1} = \frac{n}{3}$,

$x_{v_2} = \frac{n}{3}$ et $x_{v_3} = \frac{n}{3}$, nous obtenons $C_{v_1} = C_{v_2} = C_{v_3}$ et n'importe quel déplacement de charge d'un joueur depuis cette configuration ne serait pas profitable. Cet état est donc un équilibre de Nash.

La figure 2.7 en revanche montre un équilibre d'un type différent. En effet la configuration, où $x_{v_1} = \frac{n}{2}$ et $x_{v_3} = \frac{n}{2}$ tandis que les sommets v_2 et v_4 restent vides, est un équilibre de Nash. Cet équilibre se présente sous la forme d'un stable, en effet aucun des sommets chargés n'est relié par une arête. Dans le chapitre suivant nous prouverons que, pour tout jeu de placement symétrique non pondéré, les stables de taille maximale sont des équilibres de Nash et qui plus est que le stable maximum est le meilleur équilibre en termes de coût pour ce jeu.

2.3

Index des notations et conclusion

Dans ce chapitre, nous avons défini notre jeu de placement et ses différents cas d'étude. Nous avons vu que certaines variantes de ce jeu ne possèdent pas d'équilibre de Nash. Par la suite, nous nous efforçons de répondre aux questions évoquées au début de ce chapitre, à savoir s'il existait des équilibres de Nash, si tel est le cas s'ils étaient calculables facilement et dans le cas contraire la classe de complexité du jeu, pour les différents cas d'étude proposés.

Afin de simplifier la lecture du document, nous regroupons, dans le tableau 2.1, l'ensemble des notations de ces chapitres.

Notation	Description	Formule
$G = (V, A)$	le graphe orienté	
$G = (V, E)$	le graphe non orienté	
$V = \{1, \dots, m\}$	l'ensemble des sommets	
m	le nombre de sommets	
i, j ou v	un sommet	
$N^-[v]$	le voisinage intérieur fermé du sommet v	$N^-[v] = \{j : (j, v) \in A\}$
Γ_v	le voisinage ouvert du sommet v	
$N[v]$	le voisinage fermé du sommet v	$N[v] = \Gamma_v \cup \{v\}$
$(i, j) \in E$	une arête entre i et j dans le graphe $G = (V, E)$	
$(i \rightarrow j) \in A$	un arc de i vers j dans le graphe $G = (V, A)$	
$N = \{1, \dots, n\}$	ensemble des joueurs	
n	le nombre de joueurs	
ℓ	un joueur	
p_ℓ	le poids d'un joueur	
s_ℓ	la stratégie du joueur ℓ	
S	le profil de stratégies des joueurs	
x_v	la charge du sommet v	$x_v = \sum_{\ell \in N, s_\ell = v} p_\ell$
$x(S)$	la répartition de charges découlant du profil S	$x(S) = (x_1, \dots, x_n)$
D	la matrice des coefficients	
$d_{i,j} \in D$	l'impact de la charge du sommet i sur j	$d_{i,i} = 1, d_{i,j} \geq 0$
C_j	la fonction de coût du sommet j	
$C_j(S)$	le coût du sommet j suivant le profil S	$C_i(S) = \sum_{j \in N^-[i]} d_{i,j} x_j(S)$
$C^\ell(S)$	le coût du joueur ℓ suivant le profil S	$C^\ell(S) = C_{s_\ell}(S)$

Tableau 2.1 – Index des notations relatives au jeu de placement (G, D, n)

Équilibres du jeu de placement

Nous avons présenté, dans le chapitre précédent, une forme générale de notre jeu de placement. Nous avons vu que certaines variantes de ce jeu ne possèdent pas d'équilibre de Nash. Nous allons nous intéresser à la question de l'existence (ou non) d'un équilibre de Nash dans les cas d'étude du jeu de placement.

Déterminer l'existence d'un équilibre de Nash pur, sans contraintes sur les coefficients des arcs, est NP-Complet, pour les jeux asymétriques. Néanmoins, nous montrons qu'en faisant l'hypothèse que les coefficients des arcs entre chaque paire de sommets sont égaux, nous obtenons un jeu de potentiel. Considérer un tel graphe orienté ayant pour tout couple de sommets (i, j) des coefficients égaux (c.-à-d. $d_{i,j} = d_{j,i}$) revient à étudier le jeu sur un graphe non orienté ce qui nous ramène au jeu de placement symétrique. Nous montrons que ce type de jeu est bien un jeu de potentiel. Par la suite, nous nous sommes intéressés à la forme des équilibres de ce jeu et à la complexité liée à leur calcul.

Sommaire

4.1	Dynamique de meilleure réponse	79
4.2	Dynamique de réplication	80
4.2.1	Notations et algorithme <i>LRI</i>	81
4.2.2	Propriétés de l'algorithme	82
4.3	Minimisation de regret et problème de bandits manchots	83
4.3.1	Regrets et minimisation	83
4.3.2	Problème de Multi-Armed Bandit (ou Bandits Manchots)	85
4.3.3	Algorithme de minimisation de regret : <i>UCB</i>	87
4.4	Conclusion	89

En nous intéressant au cadre général du jeu de placement, nous allons montrer que l'existence d'équilibres de Nash dans un tel jeu est un problème NP-complet. Cette preuve de NP-complétude reposant principalement sur l'asymétrie de notre système, le fait de se baser sur un système symétrique permet de pallier cette difficulté. En effet, nous allons montrer par la suite que pour le jeu de placement dans un système symétrique nous avons affaire à un jeu de potentiel et qui possède donc au moins un équilibre de Nash (cf. chapitre 1). Néanmoins, nous prouverons que le calcul de ces équilibres est PLS-complet. En supprimant la contrainte de pondération des arêtes, nous montrerons que des équilibres peuvent être obtenus en temps polynomial.

Le tableau suivant résume les résultats obtenus.

	Asymétrique	Symétrique	Symétrique non pondéré
Existence d'équilibre	NP-Complet	Jeu de potentiel	Jeu de potentiel
Calcul d'équilibre	-	PLS-complet	Polynomial

3.1

Existence d'équilibres de Nash

3.1.1 Existence d'équilibres dans le jeu de placement asymétrique

Nous allons montrer que, dans le jeu de placement asymétrique, déterminer l'existence d'un équilibre de Nash pur, sans contraintes sur les coefficients des arcs, est NP-Complet. Plus formellement, voici l'énoncé du problème de décision que nous montrons tout d'abord comme étant dans NP.

Problème : Existence d'un équilibre de Nash pur sur le jeu de placement dans un graphe orienté à coefficients positifs.

Instance :

- $N = \{1, \dots, n\}$ l'ensemble des tâches (joueurs) ;
- $G = (V, A)$, graphe orienté ;
- D , la matrice de coefficients positifs ($\forall i, j \in V^2, d_{i,j} \geq 0$).

Question : Existe-t-il un équilibre de Nash pur ?


Lemme 3.1 – Existence d'un équilibre de Nash pur pour le jeu de placement asymétrique dans NP

Le problème consistant à déterminer l'existence d'équilibre de Nash pur dans un jeu asymétrique, sans contraintes sur les coefficients des arcs, appartient à la classe NP.

Preuve : L'équilibre de Nash correspondant à une configuration dans laquelle aucun joueur ne peut, de manière unilatérale, dévier de sa stratégie afin de diminuer son coût, vérifier qu'une configuration S est un équilibre de Nash s'effectue en temps polynomial.

Pour un joueur, vérifier que la stratégie choisie à l'état S est celle de moindre coût s'effectue en $O(m^2)$ opérations, m étant le nombre de sommets. En effet, calculer son coût pour un sommet donné revient à additionner, à la charge de celui-ci, la somme des charges (multipliée par le coefficient adéquat) de chacun de ses voisins. Pour le coût d'un sommet, dans le pire cas, il faut alors parcourir tous les sommets du graphe, ce qui nous donne un calcul en $O(m)$ opérations. Étant donné qu'il y a m sommets, un joueur vérifie donc bien si sa stratégie est celle de moindre coût en $O(m^2)$ opérations.

Si nous considérons cette action effectuée par les n joueurs, alors vérifier qu'une configuration est un équilibre s'effectue en $O(n \cdot m^2)$ opérations. $\square$

D'après le lemme 3.1, le problème consistant à déterminer l'existence d'un équilibre de Nash pur dans un jeu asymétrique appartient à la classe NP. Nous allons maintenant montrer que celui-ci est NP-Complet.


Théorème 3.1 – NP-Compétude de l'existence d'un équilibre de Nash pur pour le jeu de placement asymétrique

Le problème consistant à déterminer l'existence d'un équilibre de Nash pur dans un jeu asymétrique, sans contraintes sur les coefficients des arcs, est NP-Complet.

Le reste de cette section sera consacré à la preuve de ce théorème.

Nous effectuons une réduction de E3-SAT [Garey et Johnson, 1979] vers notre problème. E3-SAT consiste à déterminer la satisfiabilité d'une formule normale conjonctive donnée, dont chaque clause est formée d'exactly 3 littéraux. Soit une instance de E3-SAT ayant un ensemble de variables $U = \{u_1, u_2, \dots, u_l\}$ et un ensemble de clauses $Cl = \{cl_1, cl_2, \dots, cl_h\}$.

Nous allons montrer comment construire, en temps polynomial, un graphe orienté $G' = (V', A')$ et une matrice de coefficients D correspondant aux poids

des arcs associés à cette instance, de sorte qu'il existe un équilibre de Nash pur pour le jeu (G, D, n) si et seulement si l'instance de E3-SAT est satisfiable.

La preuve de ce théorème s'organise comme suit. Dans un premier temps, nous détaillons la construction du graphe G' . Par la suite, nous prouvons trois lemmes nous permettant de caractériser la structure d'un équilibre pour ce graphe G' . Nous montrons enfin qu'il existe une affectation permettant de passer d'une instance de E3-SAT à un équilibre de Nash pur dans G' . À l'inverse, nous montrons qu'un équilibre de Nash pur dans G' est une solution satisfaisant E3-SAT.

Construction du graphe.

Pour la construction, nous reprenons les deux exemples du chapitre précédent (exemple 1 et exemple 2 représentés respectivement dans les figures 2.3 et 2.4) caractérisant les composantes du graphe à construire.

La construction du graphe G' s'effectue de la manière suivante. Pour chacune des variables u_i , est associé un couple de sommets u_i et $\overline{u_i}$ reliés tel que sur la figure 2.3 (gadget variable). Pour chaque clause $cl_j = (u_a, u_b, u_c)$, les sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ sont créés et reliés entre eux sous forme de cycle comme dans la figure 2.4 (gadget clause). De plus, chaque sommet $v_{i,j}$ de cette clause sera relié au sommet u_i correspondant.

Ce qui nous donne $G' = (V', A')$ avec :

- $V' = \{u_i, \overline{u_i}, v_{a,j} | i \in \{1, \dots, l\}, j \in \{1, \dots, h\}, u_a \in cl_j\}$;
- $A' = \{(u_i, \overline{u_i}), (\overline{u_i}, u_i), (u_a, v_{a,j}), (v_{a,j}, v_{b,j}), (v_{b,j}, v_{c,j}), (v_{c,j}, v_{a,j}) | i \in \{1, \dots, l\}, j \in \{1, \dots, h\}, (u_a, u_b, u_c) = cl_j\} \cup \{(v, v) | v \in V'\}$;
- Nous allons définir la fonction de poids $d_{i,j} = \begin{cases} 1 & \text{si } i = j \\ k & \text{si } i \neq j \end{cases}$

De plus, nous posons $n' = 2|V'|$ le nombre de joueurs, le poids des joueurs étant unitaire. Par la suite, nous supposerons $k > 2$.

Construire le graphe G à partir d'une instance E3 – SAT se fait en temps polynomial. Notons G'_j le sous-graphe induit de G correspondant à la clause $cl_j = (u_1, \overline{u_2}, u_3)$. Pour plus de lisibilité, celui-ci sera représenté comme sur la figure 3.1 qui est un exemple sur une clause $cl_j = (u_1, \overline{u_2}, u_3)$.

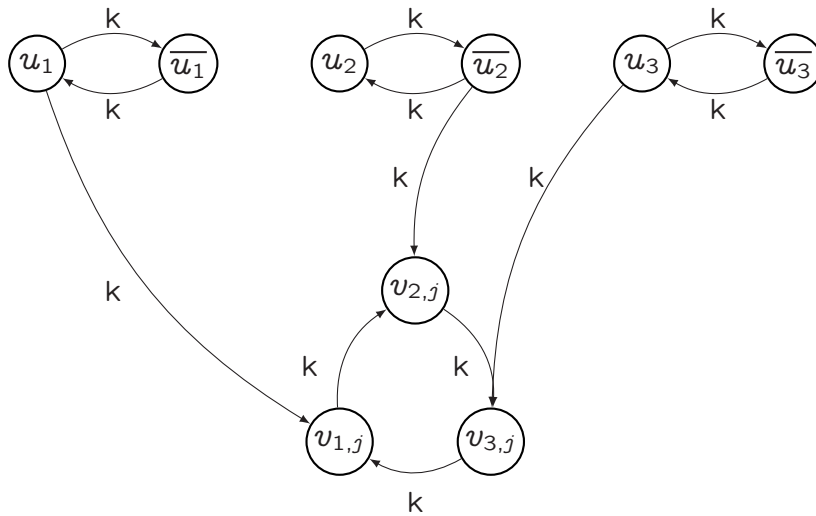


Figure 3.1 – Illustration d'une partie du jeu correspondant à la clause $cl_j = (u_1, \overline{u_2}, u_3)$

Caractérisation d'un équilibre de Nash pour G' .

Les lemmes 3.2, 3.3 et 3.4 qui suivent vont nous permettre de caractériser la structure d'un équilibre de Nash pour le graphe G' .



Lemme 3.2 – Un seul des sommets u_i ou $\overline{u_i}$ chargé

Pour chaque variable $u_i \in U$. S'il existe un équilibre de Nash, alors un seul des sommets u_i et $\overline{u_i}$ est chargé.

Preuve : Supposons qu'il existe S un équilibre de Nash. Nous allons considérer trois cas :

- **Cas 1** : les joueurs sont placés sur u_i et $\overline{u_i}$;
- **Cas 2** : les joueurs ont choisi de se placer sur un seul des sommets u_i ou $\overline{u_i}$;
- **Cas 3** : aucun des joueurs n'a choisi de se placer sur un des sommets u_i ou $\overline{u_i}$.

Cas 1. Remarquons que, d'après le lemme 2.1 ce cas n'est pas possible. En effet, aucun arc n'entre dans le gadget variable, de ce fait nous pouvons nous réduire à l'exemple 1 (figure 2.3). Un seul des sommets u_i et $\overline{u_i}$ est alors chargé.

Cas 2. La preuve de ce cas est presque identique à celle du lemme 2.1 puisque le coût des sommets u_i et $\overline{u_i}$ dépendent uniquement des charges de ces deux sommets. Pour cela, nous ne la détaillons pas.

Cas 3. Dans ce cas, comme aucun des sommets u_i et $\bar{u}_i$ n'est chargé, et comme $n' \geq 2|V'|$, il existe un sommet $v \neq u_i, \bar{u}_i$ dans G' dont la charge est au moins de 2. Cela signifie que dans S , un joueur placé sur v a un coût supérieur ou égal à 2. Un joueur placé sur v a intérêt à changer de stratégie en se plaçant sur un des sommets u_i et $\bar{u}_i$, car son coût sera égal à 1 après ce changement. Cela mène une contradiction avec le fait que S soit un équilibre de Nash. Ce cas est donc impossible.

Si S est un équilibre de Nash, alors u_i et $\bar{u}_i$ ne peuvent être simultanément chargés. □



Lemme 3.3 – Au moins un des trois sommets u_a , u_b ou u_c chargé

Soit $cl_j = (u_a, u_b, u_c)$ une clause de $Cl = \{cl_1, \dots, cl_h\}$. Si S est un équilibre de Nash, alors au moins un des trois sommets u_a , u_b et u_c de G'_j est chargé.

Preuve : Nous allons prouver ce lemme par contradiction. Supposons qu'il existe un équilibre de Nash S tel qu'aucun des trois sommets u_a , u_b et u_c de G'_j soit chargé. Soit $X_j = \{u_a, u_b, u_c, \bar{u}_a, \bar{u}_b, \bar{u}_c, v_{a,j}, v_{b,j}, v_{c,j}\}$ l'ensemble des sommets de G'_j correspondant à la clause cl_j . Le lemme 3.2 montrant que, pour tout $i \in \{a, b, c\}$, seul un des deux sommets est chargé entre u_i et $\bar{u}_i$, nous raisonnons sur la charge des sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$. Nous analysons donc les trois cas possibles suivants,

- **Cas 1 :** aucun des sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j n'est chargé ;
- **Cas 2 :** un seul joueur est placé sur les sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j ;
- **Cas 3 :** plusieurs joueurs sont placés sur les sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j .

Cas 1. Tout d'abord, nous allons considérer le cas où aucun des sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j n'est chargé. Soit S_1 cette configuration. Comme $n' \geq 2|V'|$, dans la configuration S_1 , il existe un sommet $u \in V'$ ayant une charge supérieure à 2. De ce fait, un joueur ℓ placé sur u a un coût supérieur ou égal à 2. Ce joueur ℓ a alors intérêt à changer de stratégie en se plaçant sur un des sommets $v_{a,j}$, $v_{b,j}$ ou $v_{c,j}$, car son coût sera égal à 1 s'il effectue ce changement. Cela mène une contradiction avec le fait que S_1 soit un équilibre de Nash. Ce cas est alors impossible.

Cas 2. Ensuite, nous allons considérer la configuration S_2 où un seul joueur est placé sur les sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j . Par le même argument que précédemment, il existe un sommet $u \in V'$ dont la charge est strictement supérieure à 2. Cela implique qu'un joueur ℓ placé sur u a un coût supérieur ou égal à 3. Le joueur ℓ a alors intérêt à changer de stratégie en se plaçant

sur le seul sommet chargé de $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$, car son coût sera égal à 2 s'il effectue ce changement. Cela mène une contradiction avec le fait que S_2 soit un équilibre de Nash.

Cas 3. Maintenant considérons le cas où plusieurs joueurs sont placés sur les sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j . Nous allons prouver que ce cas n'est pas possible. Pour prouver cela, la preuve est presque identique à celle du lemme 2.2 puisqu'il suffirait d'écrire toutes les équations en prenant en compte les sommets u_a , u_b et u_c et leur charge. Ici, comme nous avons supposé que pour S aucun sommet n'est chargé parmi u_a , u_b et u_c nous pouvons nous réduire au cycle formé par $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$. Le lemme 2.2 prouve que dans ce cas il n'y a pas d'équilibre de Nash.

L'étude de ces différents cas possibles nous permet donc de prouver qu'au moins un des sommets est chargé parmi u_a , u_b et u_c . $\square$



Lemme 3.4 – Au plus un des trois sommets $v_{a,j}$, $v_{b,j}$ ou $v_{c,j}$ chargé

Soit $cl_j = (u_a, u_b, u_c)$ une clause de $Cl = \{cl_1, \dots, cl_h\}$. Si S est un équilibre de Nash, alors au plus un des trois sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ de G'_j est chargé.

Preuve :

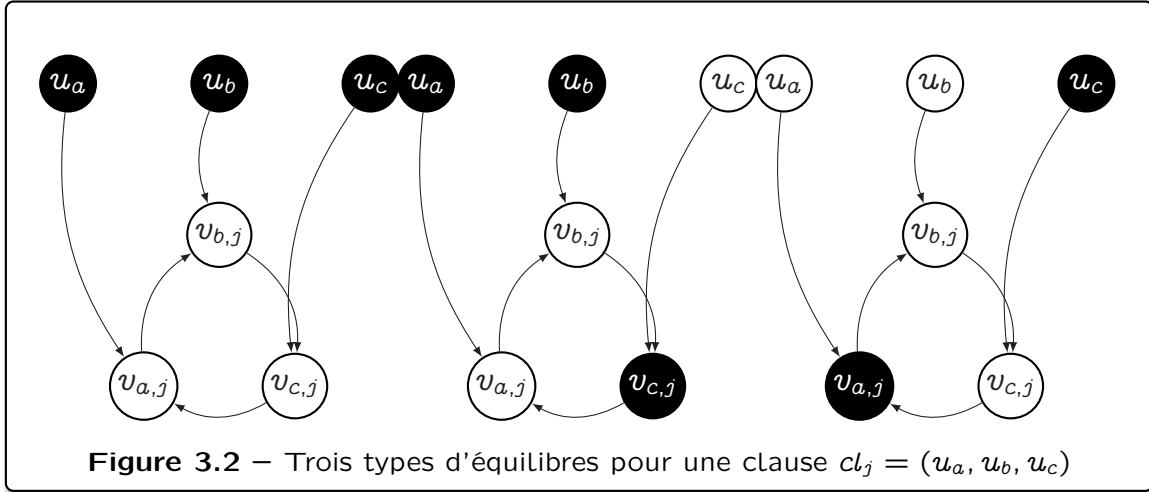
Soit $X_j = \{u_a, u_b, u_c, \overline{u_a}, \overline{u_b}, \overline{u_c}, v_{a,j}, v_{b,j}, v_{c,j}\}$ l'ensemble des sommets de G' correspondant à la clause cl_j .



Remarque

Nous allons tout d'abord prouver par contradiction que, si le sommet u_i avec $i \in \{a, b, c\}$ est chargé, alors le sommet $v_{i,j}$ ne l'est pas. Supposons que le joueur ℓ est placé sur le sommet $v_{i,j}$: $C^\ell(S) \geq kx_{u_i} + 1$ avec $k > 2$, où x_{u_i} est la charge du sommet u_i . Si le joueur ℓ décide de se placer sur le sommet u_i , alors sa charge serait de $x_{u_i} + 1$ (ceci est dû au fait que le sommet $\overline{u_i}$ n'est pas chargé, comme prouvé par le lemme 3.2).

La figure 3.2 énumère les différents cas d'équilibre. Remarquons que par le lemme 3.3, au moins un des trois sommets u_a , u_b et u_c de G'_j est chargé. Nous montrons maintenant qu'au plus un des sommets $v_{i,j}$ avec $i \in \{a, b, c\}$ est chargé. Pour cela, nous considérons les trois cas possibles dépendants du nombre de sommets chargés parmi u_a , u_b et u_c .



Cas 1. Soit S_1 un équilibre de Nash tel que les trois sommets u_a , u_b et u_c sont chargés. D'après la remarque précédente, aucun des sommets $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$ n'est chargé dans S_1 .

Cas 2. Considérons maintenant un équilibre S_2 où deux des sommets u_a , u_b et u_c sont chargés. Sans perte de généralité, supposons que u_a et u_b soient chargés. Cela implique, d'après la remarque précédente, que $v_{a,j}$ et $v_{b,j}$ ne sont pas chargés dans S_2 . Supposons que le sommet $v_{c,j}$ n'est pas chargé. Dans S_2 , il existe un sommet v dans G'_j dont la charge est supérieure à 2. Un joueur placé sur v a alors intérêt à changer de stratégie en se plaçant sur $v_{c,j}$, car son coût sera égal à 1 suite à ce changement. Cela mène une contradiction avec le fait que S_2 soit un équilibre de Nash. Donc $v_{c,j}$ doit être chargé dans S_2 .

Cas 3. Considérons S_3 un équilibre où seul un sommet des trois sommets u_a , u_b et u_c est chargé. Sans perte de généralité, supposons que u_c est chargé. Cela implique, par la remarque précédente, que $v_{c,j}$ n'est pas chargé dans S_3 . Nous pouvons prouver, en utilisant les mêmes arguments que précédemment, qu'il est impossible que les deux sommets $v_{a,j}$ et $v_{b,j}$ soient chargés dans S_3 . Supposons alors, par contradiction, que les deux sommets $v_{a,j}$ et $v_{b,j}$ sont chargés. Un joueur placé sur $v_{b,j}$ ressent alors un coût supérieur à k . Il a intérêt à changer de stratégie en se plaçant sur $v_{a,j}$, car son coût sera égal à 1 s'il effectue ce changement. Cela mène une contradiction avec le fait que S_3 soit un équilibre de Nash. Si S_3 est un équilibre, alors seul un sommet est chargé parmi $v_{a,j}$, $v_{b,j}$ et $v_{c,j}$.

Nous avons prouvé, en considérant les différents cas d'équilibre possibles, qu'au plus un des sommets $v_{i,j}$ avec $i \in \{a, b, c\}$ est chargé. $\square$

E3-SAT vers un NE pur.

Nous allons maintenant prouver que s'il existe t une affectation dans $U \rightarrow \{0, 1\}$ telle que t satisfait la formule normale conjonctive, alors le jeu (G', D, n') construit admet un équilibre de Nash pur.

Soit t une affectation $U \rightarrow \{0, 1\}$ telle que t satisfait la formule normale conjonctive. Nous allons construire un équilibre de Nash pur. Tout d'abord, nous allons sélectionner les sommets à charger, puis nous procéderons ensuite à la répartition des joueurs sur ceux-ci.

Les sommets sélectionnés pour être chargés dépendent de la valeur prise par chaque variable. Notons $\mathcal{X}$ l'ensemble de sommets qui seront chargés.

Dans un premier temps, nous sélectionnons les sommets dans le gadget variable. Pour chaque variable $u_i \in U$, nous considérons les deux cas suivants :

- si $t(u_i) = 1$ alors nous sélectionnons u_i . Donc $u_i \in \mathcal{X}$;
- sinon $\overline{u_i}$ sera sélectionné (car si $t(u_i) = 0$ cela implique $t(\overline{u_i}) = 1$). Donc $\overline{u_i} \in \mathcal{X}$.

Nous sélectionnons maintenant les sommets dans le gadget clause. Comme chaque clause est satisfaite, alors pour chacune d'elle il existe au moins un u_i tel que $t(u_i) = 1$, avec $i \in \{a, b, c\}$. Nous avons alors les trois cas possibles, dans lesquels nous avons un, deux ou trois u_i tel(s) que $t(u_i) = 1$, avec $i \in \{a, b, c\}$. Pour chacune des clauses $cl_j = (u_a, u_b, u_c) \in Cl \mid j \in \{1, \dots, h\}$ et avec $i \in (a, b, c)$ nous considérons ces trois cas, et choisissons quel(s) sommet(s) du gadget clause sélectionner afin de construire les différents cas d'équilibre illustrés par la figure 3.2.

- si $t(u_a) = t(u_b) = t(u_c) = 1$, alors nous ne sélectionnons aucun des sommets $v_{i,j}$ avec $i \in \{a, b, c\}$, du gadget clause de G'_j ;
- si nous avons deux u_i tels que $t(u_i) = 1$, avec $i \in \{a, b, c\}$, alors nous avons $t(u_{i'}) = 0$, avec $i' \in \{a, b, c\}$ et $i' \neq i$. Nous sélectionnons $v_{i',j}$ dans le gadget clause de G'_j ;
- s'il existe un unique u_i tel que $t(u_i) = 1$, avec $i \in \{a, b, c\}$, alors nous sélectionnons dans le gadget clause de G'_j le sommet $v_{i',j}$ avec $i' \in \{a, b, c\}$ tel qu'il existe un arc de $v_{i,j}$ vers $v_{i',j}$.

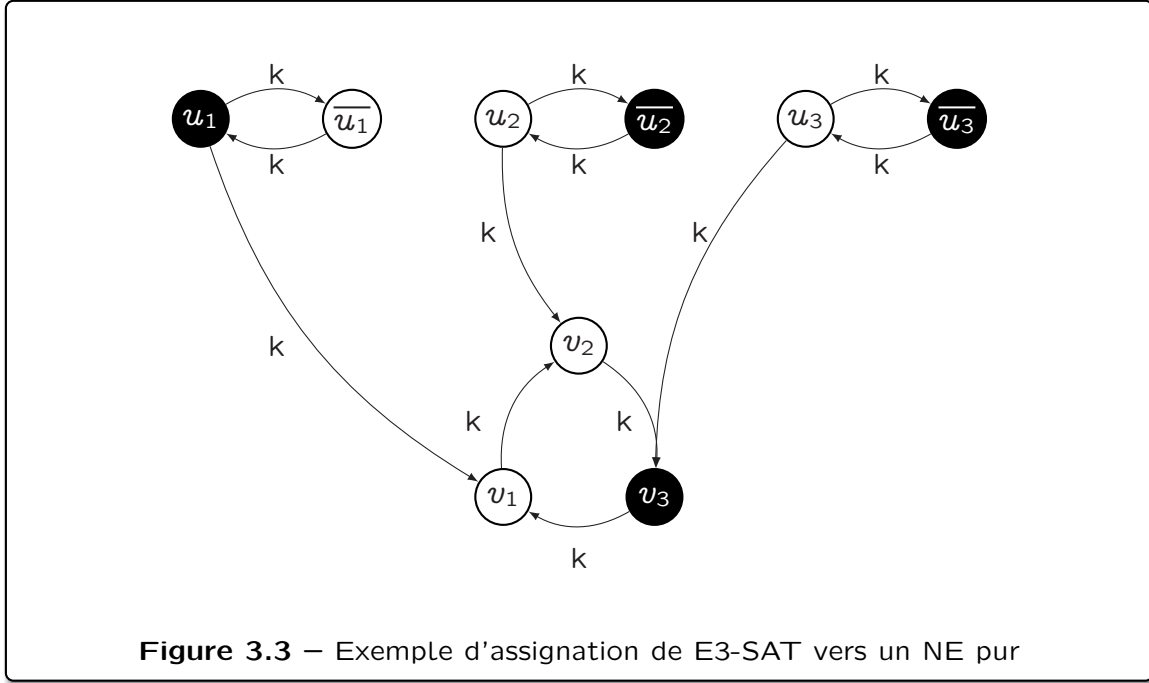
De part cette construction, nous avons, dans le graphe G' , pour chacun des littéraux toujours un seul élément chargé entre u_i et $\overline{u_i}$ d'après le lemme 2.2. En effet, $\forall i \in \{1, \dots, l\}$, si $t(u_i) = 1$ alors u_i est chargé, sinon cela implique $t(\overline{u_i}) = 1$ et donc $\overline{u_i}$ de chargé. De plus, pour chaque clause $cl_j = (u_a, u_b, u_c) \in Cl$ et avec $i \in \{a, b, c\}$, au plus un sommet $v_{i,j}$, dans G'_j le sous-graphe induit de G correspondant à la clause cl_j , est chargé. Nous obtenons trois configurations étant des équilibres d'après les trois cas du lemme 3.4.

Soit $b = |\mathcal{X}|$ le nombre de sommets à charger, alors la répartition effectuée est de $\lfloor \frac{n'}{b} \rfloor$ ou $\lfloor \frac{n'}{b} \rfloor + 1$ (selon si b divise n' ou non, et sachant $n' = 2|V'|$) sur chacun de ces sommets.

Exemple : La figure 3.3 est une assignation pour la clause $cl = (u_1, u_2, u_3)$ avec $t(u_1) = 1, t(u_2) = 0$ et $t(u_3) = 0$. Ce qui implique que nous sélectionnons

u_1 , $\overline{u_2}$ et $\overline{u_3}$. Comme $t(u_2) = 0$ et $t(u_3) = 0$ nous sélectionnons v_2 ou v_3 . Ici, nous avons sélectionné v_3 .

Nous avons $\{u_1, u_2, \overline{u_3}, v_3\}$ dans $\mathcal{X}$, l'ensemble des sommets chargés.



NE pur vers E3-SAT.

Nous montrons maintenant qu'un équilibre pur est une assignation pour une instance de E3-SAT. Cela revient à montrer qu'étant donné le graphe $G' = (V', A')$, notre représentation du jeu (G', D, n') possède au moins un des sommets u_i chargé, chaque sommet u_i représentant les variables de la formule normale conjonctive.

Soit S un équilibre de Nash pur. D'après le lemme 3.2, un seul des sommets est chargé entre u_i et $\overline{u_i}$ pour chaque variable u_i avec $i \in \{a, b, c\}$. Nous allons décrire dans un premier temps comment nous obtenons t une affectation dans $U \rightarrow \{0, 1\}$. Pour chaque variable u_i avec $i \in \{a, b, c\}$, la valeur $t(u_i)$ dépend de l'état du sommet u_i (associé à la variable u_i dans l'instance E3-SAT) et :

- si u_i est chargé alors $t(u_i) = 1$;
- sinon si $\overline{u_i}$ est chargé alors $t(u_i) = 0$.

Maintenant, pour conclure la preuve du théorème 3.1 nous allons montrer que t satisfait bien la formule normale conjonctive. Il nous faut prouver que chacune des clauses de Cl est satisfaite.

Nous nous intéressons à une clause $cl_j = (u_a, u_b, u_c)$.

Pour cela, nous allons nous focaliser sur G'_j , le graphe induit de G' pour la clause $cl_j = (u_a, u_b, u_c)$.

D'après le lemme 3.3, si aucune des variables u_i , avec $i \in \{1, 2, 3\}$, n'est chargée dans le graphe alors la configuration n'est pas un équilibre. Cela signifie qu'il existe au moins un littéral x de la clause dont $t(x) = 1$, la clause est alors satisfaite. La formule normale conjonctive est bien satisfaite par t . Un équilibre de Nash satisfait donc bien la formule normale conjonctive.

Ceci conclut la preuve du théorème 3.1.

La preuve de NP-complétude repose essentiellement sur le fait que le graphe soit asymétrique. Nous allons maintenant nous focaliser sur le graphe symétrique et voir que celui-ci est un jeu de potentiel.

3.1.2 Équilibres de Nash dans le jeu de placement symétrique

Dans le chapitre précédent nous avons défini le jeu de placement symétrique comme étant le jeu dans un graphe non orienté $G = (V, E)$ où chaque arête entre deux sommets i et j a un poids $d_{i,j} = d_{j,i}$ correspondant à l'impact de la charge du sommet i sur le coût de j (et inversement). Dans ce qui suit, nous allons montrer des résultats sur l'existence des équilibres de Nash, dans un premier temps dans le cas atomique du jeu de placement, puis dans le cas non atomique.

3.1.2.1 Contexte atomique : existence d'une fonction de potentiel

Nous allons montrer, grâce à une fonction de potentiel, que ce jeu est un jeu de potentiel et admet donc des équilibres de Nash purs.



Théorème 3.2 – Existence d'une fonction de potentiel

Étant donné un graphe non orienté $G = (V, E)$, dont toute arête $(i, j) \in E$ a un poids $d_{i,j} = d_{j,i}$ et $d_{i,i} = 1$, le jeu pondéré symétrique dans lequel chaque joueur ℓ cherche à minimiser

$$C^\ell(x_1, \dots, x_m) = \sum_{j \in N[i]} d_{j,i} x_j(S),$$

est un jeu de potentiel pondéré.

Il existe une fonction de potentiel Φ telle que si d'un profil S à S' un joueur ℓ migre d'un sommet i vers j alors :

$$2(C^\ell(S) - C^\ell(S')) = \Phi(S) - \Phi(S')$$

avec $C^\ell(S) = C_{s_\ell}(S) = \sum_{v \in N[s_\ell]} x_v(S)$, pour $S = (s_1, \dots, s_\ell, \dots, s_n)$ et

$$\Phi(S) = C_{moyen}(S) = \sum_{v \in V} x_v(S) \cdot C_v \quad (3.1)$$

Preuve : Rappelons que $\forall v \in V, w \in V, d_{v,w} = d_{w,v}$ et que si $(v, w) \notin E, d_{v,w} = 0$. De plus, nous avons $\forall v \in V d(v, v) = 1$. Pour chaque joueur $\ell \in N$, nous considérons les poids $p_\ell = \frac{1}{n}$ et non $p_\ell = 1$, comme utilisé dans le contexte atomique, afin de ne pas surcharger les calculs. En effet, par la suite nous considérons le coût moyen des joueurs, ayant n joueurs cela permet de ne pas faire apparaître le facteur $\frac{1}{n}$ dans les calculs.

Supposons une configuration S quelconque avec $x(S) = (x_1, x_2, \dots, x_m)$ la répartition des joueurs sur les sommets.

Soit S' une configuration issue de S telle que seules les charges x_1 et x_2 aient été modifiées. Supposons alors, sans perte de généralité, un transfert de charges d'un joueur ℓ du sommet 1 vers 2. Cela implique que nous avons en $S', x(S') = (x'_1, x'_2, \dots, x'_m)$ avec

$$\begin{cases} x'_1 = x_1 - t \\ x'_2 = x_2 + t \\ \forall i \geq 2, x'_i = x_i \end{cases} \quad (3.2)$$

et t tel que les contraintes sur le vecteur $x, \forall v \in V, x_v \geq 0$ et $\sum_{v \in V} x_v = 1$ soient vérifiées, soit $-x_2 \leq t \leq x_1$.

Posons $\Delta_{1,2} = C_2(S') - C_1(S)$, la différence de coût entre le sommet source (1) avant migration et le sommet destination (2) après migration.

Ce transfert est profitable aux joueurs ayant migré si $\Delta_{1,2} < 0$. Or

$$\begin{aligned} \Delta_{1,2} &= C_2(S') - C_1(S) \\ &= \sum_{v \in V} d_{2,v} \cdot x'_v - \sum_{v \in V} d_{1,v} \cdot x_v \\ &= \sum_{v > 2} d_{2,v} \cdot x'_v - \sum_{v > 2} d_{1,v} \cdot x'_v + d_{2,2} \cdot x'_2 - d_{1,2} \cdot x_2 + d_{2,1} \cdot x'_1 - d_{1,1} \cdot x_1 \end{aligned}$$

D'après la formule (3.2), nous avons

$$\begin{aligned} \Delta_{1,2} &= \sum_{v > 2} (d_{2,v} - d_{1,v}) \cdot x_v + d_{2,2} \cdot (x_2 + t) - d_{1,2} \cdot x_2 + d_{2,1} \cdot (x_1 - t) - d_{1,1} \cdot x_1 \\ &= \sum_{v \in V} (d_{2,v} - d_{1,v}) \cdot x_v + (d_{2,2} - d_{2,1}) \cdot t \end{aligned}$$

Nous avons calculé $\Delta_{1,2}$, la différence de coût ressentie par le joueur ℓ suite à sa migration. Nous allons maintenant montrer que cette différence de coût est

proportionnelle à la différence de potentiel entre S et S' . Nous considérons comme fonction de potentiel la formule du coût moyen d'un joueur, étant donnée la configuration S , que nous avons définie comme suit

$$\begin{aligned}
 C_{moyen}(S) &= \sum_{v \in V} x_v \cdot C_v \\
 &= \sum_{v \in V} x_v \sum_{w \in N[v]} d_{v,w} \cdot x_w \\
 &= 2 \sum_{(v,w) \in E} d_{v,w} (x_v \cdot x_w) + \sum_{v \in V} d_{v,v} \cdot x_v^2 \\
 &= 2 \sum_{(v,w) \in E} d_{v,w} (x_v \cdot x_w) + \sum_{v \in V} x_v^2
 \end{aligned}$$

Nous pouvons maintenant en déduire le coût moyen en S' .

$$\begin{aligned}
 C_{moyen}(S') &= 2 \sum_{(v,w) \in E} d_{v,w} (x'_v \cdot x'_w) + \sum_{v \in V} (x'_v)^2 \\
 &= 2 \sum_{\substack{(v,w) \in E \\ v > 2, w > 2}} d_{v,w} (x'_v \cdot x'_w) + \sum_{v > 2} (x'_v)^2 + (x'_1)^2 + (x'_2)^2 + 2d_{1,2} (x'_1 \cdot x'_2) \\
 &\quad + 2 \sum_{v > 2} d_{1,v} (x'_1 \cdot x_v) + 2 \sum_{v > 2} d_{2,v} (x'_2 \cdot x_v) \\
 &= 2 \sum_{\substack{(v,w) \in E \\ v > 2, w > 2}} d_{v,w} (x_v \cdot x_w) + \sum_{v > 2} (x_v)^2 + (x_1 - t)^2 + (x_2 + t)^2 \\
 &\quad + 2d_{1,2} (x_1 - t)(x_2 + t) + 2 \sum_{v > 2} d_{1,v} (x_1 - t)x_v + 2 \sum_{v > 2} d_{2,v} (x_2 + t)x_v
 \end{aligned}$$

Nous avons calculé le coût moyen pour nos deux états, S avant migration et S' l'état obtenu après migration d'un unique joueur. Nous pouvons maintenant calculer la différence entre $C_{moyen}(S')$ et $C_{moyen}(S)$.

$$\begin{aligned}
 C_{moyen}(S') - C_{moyen}(S) &= 2 \sum_{\substack{(v,w) \in E \\ v > 2, w > 2}} d_{v,w} (x_v \cdot x_w) - 2 \sum_{\substack{(v,w) \in E \\ v > 2, w > 2}} d_{v,w} (x_v \cdot x_w) \\
 &\quad + \sum_{v > 2} (x_v)^2 - \sum_{v > 2} (x_v)^2 + (x_1 - t)^2 + (x_2 + t)^2 \\
 &\quad - x_1^2 - x_2^2 + 2d_{1,2} (x_1 - t)(x_2 + t) - 2d_{1,2} (x_1 \cdot x_2) \\
 &\quad + 2 \sum_{v > 2} d_{1,v} (x_1 - t)x_v - 2 \sum_{v > 2} d_{1,v} x_1 \cdot x_v \\
 &\quad + 2 \sum_{v > 2} d_{2,v} (x_2 + t)x_v - 2 \sum_{v > 2} d_{2,v} x_2 \cdot x_v
 \end{aligned}$$

Nous obtenons alors

$$\begin{aligned}
C_{\text{moyen}}(S') - C_{\text{moyen}}(S) &= 2t^2 - 2d_{1,2}t^2 - 2tx_1 + 2tx_2 + 2d_{1,2}x_1t - 2d_{1,2}x_2t \\
&\quad - 2t \sum_{v>2} d_{1,v}x_v + 2t \sum_{v>2} d_{2,v}x_v \\
&= 2t^2(1 - d_{1,2}) + 2t(\underbrace{-x_1}_{=-d_{1,1}x_1} - d_{1,2}x_2 - \sum_{v>2} d_{1,v}x_v \\
&\quad + d_{2,1}x_1 + \underbrace{x_2}_{=d_{2,2}x_2} + \sum_{v>2} d_{2,v}x_v) \\
&= 2t^2(1 - d_{1,2}) + 2t \sum_{v \in V} (d_{2,v} - d_{1,v})x_v \\
&= 2t\Delta_{1,2}
\end{aligned}$$

La fonction de coût moyen est bien une fonction de potentiel. En effet, nous avons montré que la différence de potentiel entre deux configurations est proportionnelle à la différence des coûts du joueur ayant migré. $\square$

Nous avons prouvé que le jeu de placement symétrique est un jeu de potentiel et, de ce fait, admet au moins un équilibre de Nash pur. Nous nous intéressons maintenant au contexte non atomique de ce jeu et à l'existence d'équilibres dans ce cadre.

3.1.2.2 Contexte non atomique



Théorème 3.3 – Existence d'équilibre non atomique

Soit D une matrice réelle carrée de dimension $m \geq 1$, dont les éléments sont positifs, correspondant à la matrice d'adjacence du graphe. Le jeu de placement admet un équilibre non atomique.

Preuve : Nous reprenons le principe de la preuve d'existence d'un équilibre de Nash symétrique.

Rappelons que m est le nombre de sommets du graphe et que nous notons $M = \{1, \dots, m\}$ l'ensemble des sommets. Comme D est la matrice d'adjacence, le coût associé au sommet i peut s'écrire sous la forme de produit matrice vecteur

$$C_i(x) = D x_i$$

avec $x \in \Sigma_m$ une répartition de charges.

Nous définissons une fonction F de Σ_m dans Σ_m par

$$F_i(x) = \frac{x_i + \sum_{j \in M} x_j \cdot \max(0, C_j(x) - C_i(x))}{1 + \sum_{j, l \in M^2} x_j \cdot \max(0, C_j(x) - C_l(x))}$$

pour $i \in \{1, \dots, m\}$.

Nous allons tout d'abord montrer que F est à valeurs dans le simplexe Σ_m . Nous pouvons remarquer que pour tout $i \in \{1, \dots, m\}$, $F_i(x)$ est bien définie et vaut au plus 1. Tout d'abord nous allons prouver que $\forall i \in M, 0 \leq F_i(x) \leq 1$. Comme $F_i(x)$ fait intervenir une somme entre 1 et un maximum entre 0 et une différence de coût de deux sommets au numérateur et dénominateur, il est clair que $F_i(x) \geq 0$. De plus, comme $\sum_{i \in M} x_i = 1$ alors $x_i + \sum_{j \in M} x_j \cdot \max(0, C_j(x) - C_i(x)) \leq 1 + \sum_{j, l \in M^2} x_j \cdot \max(0, C_j(x) - C_l(x))$. Nous avons alors bien $\forall i \in M, 0 \leq F_i(x) \leq 1$.

Nous cherchons maintenant à prouver que $\sum_{i \in M} F_i(x) = 1$.

$$\begin{aligned} \sum_{i \in M} F_i(x) &= \frac{\sum_{i \in M} (x_i + \sum_{j \in M} x_j \cdot \max(0, C_j(x) - C_i(x)))}{1 + \sum_{j, l \in M^2} x_j \cdot \max(0, C_j(x) - C_l(x))} \\ &= \frac{1 + \sum_{i, j \in M^2} x_j \cdot \max(0, C_i(x) - C_j(x))}{1 + \sum_{j, l \in M^2} x_j \cdot \max(0, C_j(x) - C_l(x))} \\ &= 1 \end{aligned}$$

Comme le maximum de deux fonctions continues est continu et que la somme ainsi que le produit de fonctions continues sont continus, nous pouvons en déduire que le dénominateur et le numérateur de F_i sont continus. De plus, le dénominateur est strictement positif, cela implique que la fonction F_i est continue pour $i \in \{1, \dots, m\}$.

La fonction F est bien de Σ_m dans Σ_m et continue.

F admet un point fixe par le théorème de Brouwer dont nous rappelons l'énoncé ci-dessous.



Théorème 3.4 – Théorème de Brouwer [Brouwer,]

Toute application continue d'une partie convexe compacte non vide de $\mathbb{R}^n$ dans elle-même possède un point fixe.

Maintenant, il suffit de remarquer que Σ_m est convexe et compacte. D'après ce théorème, il existe $x \in \Sigma_m$ tel que

$$x_i = \frac{x_i + \sum_{j \in M} x_j \cdot \max(0, C_j(x) - C_i(x))}{1 + \sum_{j, l \in M^2} x_j \cdot \max(0, C_j(x) - C_l(x))} \quad (3.3)$$

pour tout $i \in \{1, \dots, m\}$.

Soit x un tel point fixe et soit $i \in M$ tel que $c_i(x)$ soit maximal parmi les $i \in M$ pour lesquels $x_i > 0$. Alors pour tout $j \in M$,

$$x_j \cdot \max(0, C_j(x) - C_i(x)) = 0$$

puisque si $C_j(x) > C_i(x)$, alors $x_j = 0$. Nous en déduisons que la somme du numérateur de (3.3) est nulle, et donc aussi que

$$\sum_{j,l \in M^2} x_j \cdot \max(0, C_j(x) - C_l(x)) = 0$$

autrement dit, pour tous $l, j \in M^2$, $x_j > 0$. Cela implique $C_j(x) \leq C_l(x)$, ce qui est exactement la définition d'un équilibre non atomique. $\square$

Nous avons montré que dans un jeu de placement symétrique, contrairement au jeu asymétrique nous pouvons prouver l'existence d'équilibres aussi bien dans un cadre atomique que non atomique. Néanmoins, prouver l'existence d'équilibres dans un jeu ne nous garantit pas que ceux-ci soient facilement calculables. Dans ce qui suit, nous nous intéressons au calcul de ces équilibres dans le jeu de placement symétrique.

3.2

Calcul des équilibres de Nash

L'existence d'équilibres dans le jeu de placement asymétrique étant incertaine, nous ne nous intéressons pas à la complexité de leur calcul. Par la suite, l'étude de la structure des équilibres dans le jeu de placement symétrique non pondéré nous permettra de montrer que des équilibres peuvent être calculés en temps polynomial dans ce jeu. Nous nous intéressons donc ici uniquement au cas du jeu de placement symétrique sans restriction sur le poids des arêtes du graphe.

Nous cherchons à prouver que dans le jeu de placement symétrique, sur un graphe non orienté $G = (V, E)$ où chaque arête entre deux sommets i et j a un poids quelconque, trouver un équilibre est PLS-complet. Pour ce faire, nous utilisons une réduction à partir de Max-Cut, et plus précisément Max-Cut

avec un voisinage *FLIP*. Nous définissons ci-dessous Max-Cut/*FLIP*. Pour une définition plus générale du problème Max-Cut, se reporter au chapitre 5.



Définition 3.1 – Max-Cut avec voisinage FLIP

Le problème Max-Cut avec le voisinage *FLIP* est défini comme suit :

- Entrée : $G = (V, E)$ un graphe pondéré avec la fonction poids $w : E \mapsto \mathbb{N}$;
- Solutions faisables : une coupe correspondant à une partition en 2 ensembles A et B de l'ensemble de sommets V ;
- Fonction objectif : la valeur de la coupe ;
- Voisinage d'une solution : deux coupes sont voisines si l'on peut obtenir l'une en déplaçant uniquement un sommet d'un ensemble à l'autre.



Théorème 3.5 – PLS-complétude de Max-Cut/*FLIP* [Schaffer, 1991]

Le problème Max-Cut avec le voisinage *FLIP* est *PLS*-complet même si les poids sont négatifs.

Grâce à ce théorème sur la *PLS*-complétude de Max-Cut avec voisinage *FLIP*, nous pouvons démontrer que trouver un équilibre de Nash dans notre jeu de placement symétrique est *PLS*-complet.

Tout d'abord, nous prouvons que trouver un équilibre de Nash dans notre jeu de placement symétrique est dans *PLS*. Nous allons définir dans un premier temps la notion de voisinage pour ce jeu.

Soient S et S' deux profils de stratégies et I une instance de notre jeu. Les deux profils S et S' sont dits voisins si S' peut être obtenu à partir de S (et inversement) en modifiant la stratégie d'un seul joueur. Plus formellement nous avons $S = \{s_i, \dots, s_i, \dots, s_n\}$ et $S' = \{s_i, \dots, s'_i, \dots, s_n\}$, seule la stratégie du joueur i diffère entre S et S' , les stratégies des autres joueurs restent inchangées.

Maintenant, afin de voir ce problème comme un problème d'optimisation il nous suffit de voir que la fonction Φ du théorème 3.2 a la propriété suivante :

$$\Phi(S) \text{ est un minimum local} \Leftrightarrow S \text{ est un équilibre de Nash pur}$$


Théorème 3.6 – Calcul d'équilibre de Nash dans le jeu de placement symétrique

Le problème consistant à calculer un équilibre de Nash pur dans un jeu symétrique, sans contraintes sur les coefficients des arêtes, est PLS-complet.

Preuve : Soit I une instance du Max-Cut avec voisinage *FLIP* correspondant au graphe $G = (V, E)$, dont les arêtes sont pondérées par la fonction w . Nous allons construire la fonction σ qui calcule un graphe $G' = (V', E')$, ayant une fonction de poids sur les arêtes $d : E \mapsto \mathbb{Q}$, à partir de I de la façon suivante :

- À chaque sommet u de V sont associés deux sommets u_A et u_B dans V' et une arête (u_A, u_B) dans $E' : V' = \{u_A, u_B | u \in V\}$;
- À chaque arête $e = (u, z)$ de E sont associées les arêtes (u_A, z_B) , (u_A, z_A) , (u_B, z_A) , (u_B, z_B) et (u_A, u_B) dans $E' : E' = \{(u_A, z_B), (u_A, z_A), (u_B, z_A), (u_B, z_B), (u_A, u_B) | (u, v) \in E, u \in V\}$;
- Nous fixons les poids d de chacune de ces arêtes :

$$\begin{aligned} d_{u_A, u_B} &= 1 & \forall u \in V \\ d_{u_A, z_B} = d_{u_B, z_A} &= \frac{1}{(\mathcal{W}+1)\Delta} - \frac{w(u, z)}{\Delta(\mathcal{W}+1)^2} & \forall (u, z) \in E \\ d_{u_A, z_A} = d_{u_B, z_B} &= \frac{1}{(\mathcal{W}+1)\Delta} & \forall (u, z) \in E \end{aligned}$$

avec $\mathcal{W} = \max\{w(e) | e \in E\}$ et Δ le degré maximum de G ;

- $|V|$ joueurs devant se placer sur le graphe G' .

La figure 3.4 illustre la construction du graphe G' à partir de G .

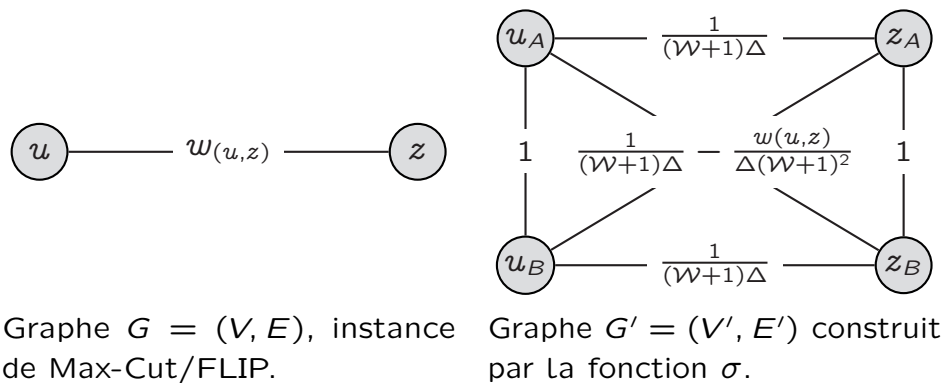


Figure 3.4 – Construction du graphe G' à partir de G

Cette fonction σ calcule la nouvelle instance en temps polynomial par rapport à la taille de l'instance du problème Max-Cut.

Nous avons construit G' de telle manière que les coupes localement maximales du graphe correspondent à un équilibre de Nash. Le reste de cette preuve est consacré à prouver qu'une coupe locale ainsi construite est bien un équilibre et inversement.

Coupe localement maximale vers équilibre de Nash.

Soit (A, B) une coupe localement maximale de G . Nous allons construire un équilibre de Nash à partir de (A, B) .

Chaque sommet u de V est associé à un joueur qui peut se placer soit sur u_A ou sur u_B en fonction de l'appartenance de u dans la coupe :

- si u est dans A alors le sommet u_A est chargé ;
- sinon c'est le sommet de B , soit u_B qui est chargé.

Nous allons maintenant prouver que ce placement correspond à un équilibre de Nash.

Soit u un sommet de V . Sans perte de généralité, nous supposons que u est dans A . Le coût C_{u_A} du joueur placé sur u_A est

$$C_{u_A} = 1 + \sum_{z \in \Gamma_{G(u) \cap A}} d_{u_A, z_A} + \sum_{z \in \Gamma_{G(u) \cap B}} d_{u_A, z_B}$$

Cela nous donne

$$\begin{aligned} C_{u_A} &= 1 + \sum_{z \in \Gamma_{G(u) \cap A}} \frac{1}{(\mathcal{W} + 1)\Delta} + \sum_{z \in \Gamma_{G(u) \cap B}} \frac{1}{(\mathcal{W} + 1)\Delta} - \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2} \\ &= 1 + \frac{|\Gamma_{G(u)}|}{\Delta(\mathcal{W} + 1)} - \sum_{z \in \Gamma_{G(u) \cap B}} \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2} \end{aligned}$$

Maintenant, nous allons prouver que le joueur placé sur u_A n'a pas intérêt à changer sa stratégie. Tout d'abord, nous pouvons constater que $C_{u_A} < 2$. Si ce joueur veut se placer sur un sommet z_A ou z_B avec $z \neq u$ alors son coût serait supérieur à 2 étant donné qu'il existe déjà un joueur sur z_A ou sur z_B (car $d_{z_A, z_B} = 1$). Si ce joueur veut se déplacer sur le sommet u_B , alors son nouveau coût C'_{u_B} sera

$$C'_{u_B} = 1 + \frac{|\Gamma_{G(u)}|}{\Delta(\mathcal{W} + 1)} - \sum_{z \in \Gamma_{G(u) \cap A}} \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2}$$

Nous allons maintenant calculer la différence $C_{u_A} - C'_{u_B}$ afin de déterminer si le joueur a intérêt à se déplacer.

$$C_{u_A} - C'_{u_B} = \sum_{z \in \Gamma_{G(u) \cap A}} \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2} - \sum_{z \in \Gamma_{G(u) \cap B}} \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2} \quad (3.4)$$

Rappelons que (A, B) est une coupe localement maximale. Cela signifie que comme u est dans A , on a $\sum_{z \in \Gamma_{G(u) \cap B}} w(u, z) \geq \sum_{z \in \Gamma_{G(u) \cap A}} w(u, z)$.

En utilisant la formule (3.4), on obtient que $C_{u_A} \leq C'_{u_B}$. Ceci signifie que le joueur placé sur u_A n'a pas intérêt à changer de stratégie. En conclusion, si (A, B) est une coupe localement maximale, alors le placement précédemment décrit est un équilibre de Nash.

Équilibre de Nash vers une coupe localement maximale.

Inversement, considérons S un équilibre de Nash. Notons $\mathcal{J}$ les sommets chargés dans G' .

Nous allons montrer par contradiction qu'un seul des sommets u_A et u_B est chargé pour chaque sommet u de V .

Supposons que u_A et u_B sont simultanément chargés. Par conséquent, la charge de ces deux sommets est supérieure à 2.

De plus, par le principe du « pigeon hole », cela signifie qu'il existe un sommet z de V tel que deux sommets z_A et z_B ne sont pas chargés. Considérons le profil S' , issu du profil S , tel que le joueur se trouvant sur le sommet u_A se place désormais sur z_A . Majorons alors la charge du sommet z_A pour S'

$$C_{z_A} = 1 + \sum_{x \in \Gamma_G(z)} \frac{1}{\Delta(\mathcal{W} + 1)} \leq 1 + \frac{1}{(\mathcal{W} + 1)} < 2$$

avec Δ le degré maximum du graphe G .

Ceci induit une contradiction avec le fait que S est un équilibre de Nash. Donc pour chaque sommet u de G , les sommets u_A et u_B de G' ne sont pas simultanément chargés. Cela implique que pour chaque sommet u de G , un seul des sommets u_A ou u_B est chargé.

Maintenant, nous allons construire une coupe locale (A, B) dans G en fonction de S l'équilibre de Nash. Un sommet u est dans A si le sommet u_A est chargé dans S sinon il est dans B . Il reste à vérifier si la coupe ainsi construite est bien un maximum local.

Soit u un sommet. Sans perte de généralité, nous supposons que u est dans A . Le coût de u pour la coupe est

$$\sum_{z \in \Gamma_{G(u) \cap B}} w(u, z)$$

Comme S est un équilibre, cela signifie que le joueur placé sur u_A n'a pas intérêt à changer de stratégie et donc de se placer sur u_B . Cela signifie que

$$1 + \frac{|\Gamma_{G(u)}|}{\Delta(\mathcal{W} + 1)} - \sum_{z \in \Gamma_{G(u)} \cap A} \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2} \geq 1 + \frac{|\Gamma_{G(u)}|}{\Delta(\mathcal{W} + 1)} - \sum_{z \in \Gamma_{G(u)} \cap B} \frac{w(u, z)}{\Delta(\mathcal{W} + 1)^2} \quad (3.5)$$

Par simplification de l'inéquation précédente, nous obtenons

$$\sum_{z \in \Gamma_{G(u)} \cap A} w(u, z) \leq \sum_{z \in \Gamma_{G(u)} \cap B} w(u, z)$$

Donc cela signifie que (A, B) est bien un maximum local. □

Nous avons montré que le calcul d'un équilibre de Nash dans le jeu de placement symétrique est PLS-complet si nous ne faisons aucune restriction sur le poids des arêtes du graphe. Dans ce qui suit, nous effectuons donc des hypothèses restrictives sur ces poids afin de déterminer la structure des équilibres de Nash dans ces cas particuliers.

3.3

Structure des équilibres de Nash

Dans la première section de ce chapitre, nous avons prouvé que notre jeu de placement dans son cadre général était NP-complet. De ce fait, l'existence d'équilibres étant incertaine, nous ne nous sommes pas intéressés à ses caractéristiques.

De par sa fonction de potentiel, le jeu de placement symétrique dispose d'au moins un équilibre de Nash (cf. chapitre 1). De plus, nous avons prouvé la complexité de calcul de ces équilibres. Nous allons, dans ce qui suit, nous focaliser dans un premier temps sur les jeux de placement symétriques pondérés, où nous prouverons quelques résultats sur la structure des équilibres qui nous mettront sur la piste du meilleur équilibre dans les jeux symétriques non pondérés.

3.3.1 Équilibres dans des cas particuliers de jeu de placement symétrique

Afin de déterminer la forme et la complexité du calcul des équilibres dans le jeu pondéré non orienté, nous nous focalisons dans un premier temps sur la recherche d'équilibres non atomiques dans un cas particulier. Nous nous plaçons dans un jeu de placement symétrique sur un graphe non orienté $G = (V, E)$ où chaque arête entre deux sommets i et j a un poids $d_{i,j} = \frac{1}{2}$. Dans la section suivante nous verrons le cas non pondéré, soit tel que $\forall(i, j) \in E, d_{i,j} = d_{j,i} = 1$, où de nombreuses propriétés ont été observées.

Nous considérons, pour le jeu de placement symétrique pondéré, deux types de jeux. Nous établissons la structure des équilibres dans un premier temps dans un jeu non atomique sur une chaîne, puis dans un jeu atomique sur un graphe quelconque.

3.3.1.1 Équilibres non atomiques sur une chaîne

Nous rappelons qu'afin d'uniformiser les notations nous considérons m comme étant le nombre de sommets du graphe, n étant le nombre de joueurs. Nous nous plaçons ici dans un jeu de placement symétrique sur une chaîne pondérée où chaque arête entre deux sommets i et j de cette chaîne a un poids $d_{i,j} = \frac{1}{2}$.

La figure 3.5 représente les trois types d'équilibres possibles sur une chaîne de m sommets dont chacune des arêtes a un poids égal à $\frac{1}{2}$.

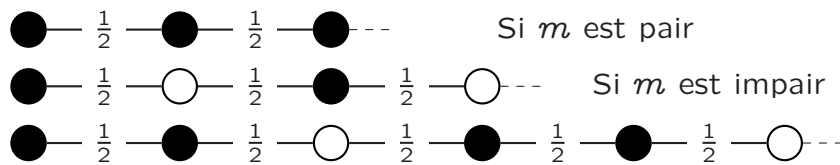


Figure 3.5 – Équilibres non atomiques sur une chaîne pondérée de m sommets ($\forall(i, j) \in E, d_{i,j} = \frac{1}{2}$). La dernière chaîne représente un équilibre ne dépendant pas de la parité de m . Les sommets en noirs sont les sommets chargés, ceux en blanc sont vides.

Nous montrons dans un premier temps, dans le lemme 3.5, qu'il existe un équilibre où tous les sommets sont chargés si le nombre de sommets de la chaîne est pair, ou bien où les sommets chargés forment un stable maximum si le nombre de sommets est impair. Nous montrerons par la suite, dans le lemme 3.6 que dans un graphe quelconque, il existe un équilibre où les sommets chargés forment un stable tel que chaque sommet non chargé est voisin d'au moins deux sommets du stable.

 **Lemme 3.5 – Équilibres non atomiques dans une chaîne pondérée**
 $(\forall (i, j) \in E, d_{i,j} = \frac{1}{2})$

Soit une chaîne de m sommets dont chacune des arêtes a un poids égal à $\frac{1}{2}$, avec $m \geq 1$. La répartition de charges suivante est un équilibre de Nash pur :

$$\begin{cases} x_{2k} = kx_2 \\ x_{2k+1} = x_1 - kx_2 \end{cases} \quad (3.6)$$

avec

- si m est pair, alors $m = 2q \Rightarrow \begin{cases} x_1 = qx_2 \\ x_2 = \frac{1}{(\frac{m}{2}+1)q} \end{cases}$
- si m est impair, alors $m = 2q + 1 \Rightarrow \begin{cases} x_1 = \frac{1}{\frac{m}{2}} \\ x_2 = 0 \end{cases}$

Preuve : Soit une chaîne de m sommets dont chacune des arêtes a un poids égal à $\frac{1}{2}$, avec $m \geq 1$. Supposons un équilibre où tous les sommets sont chargés. D'après sa définition, à l'équilibre chaque joueur ressent le même coût. L'ensemble des sommets étant chargés le coût de chaque sommet est identique. Rappelons que le coût associé à un sommet v est noté C_v .

Nous voulons montrer que la solution de la formule 3.6 est bien un équilibre, et donc satisfait $C_1 = C_2 = \dots = C_m$, avec

$$\begin{cases} C_1 = x_1 + \frac{1}{2}x_2 \\ C_2 = \frac{1}{2}x_1 + x_2 + \frac{1}{2}x_3 \\ \dots \\ C_i = \frac{1}{2}x_{i-1} + x_i + \frac{1}{2}x_{i+1} \\ \dots \\ C_{m-1} = \frac{1}{2}x_{m-2} + x_{m-1} + \frac{1}{2}x_m \\ C_m = \frac{1}{2}x_{m-1} + x_m \end{cases}$$

Nous pouvons alors en déduire x_{2k} , depuis $C_{2k-2} = C_{2k-1}$.

$$C_{2k-2} = C_{2k-1} \Leftrightarrow \frac{1}{2}x_{2k-3} + x_{2k-2} + \frac{1}{2}x_{2k-1} = \frac{1}{2}x_{2k-2} + x_{2k-1} + \frac{1}{2}x_{2k}$$

$$\Leftrightarrow x_{2k-3} + x_{2k-2} = x_{2k-1} + x_{2k}$$

$$\Leftrightarrow x_1 - (k-2)x_2 + (k-1)x_2 = x_1 - (k-1)x_2 + x_{2k} \text{ d'après Eq. (3.6)}$$

$$\Leftrightarrow x_{2k} = kx_2$$

De la même manière, nous pouvons déduire x_{2k+1} , depuis $C_{2k-1} = C_{2k}$.

$$\begin{aligned}
 C_{2k-2} = C_{2k-1} &\Leftrightarrow \frac{1}{2}x_{2k-3} + x_{2k-2} + \frac{1}{2}x_{2k-1} = \frac{1}{2}x_{2k-1} + x_{2k} + \frac{1}{2}x_{2k+1} \\
 &\Leftrightarrow x_{2k-2} + x_{2k-1} = x_{2k} + x_{2k+1} \\
 &\Leftrightarrow (k-1)x_2 + x_1 - (k-1)x_2 = kx_2 + 2x_{k+1} \text{ d'après Eq. (3.6)} \\
 &\Leftrightarrow x_{2k+1} = x_1 - kx_2
 \end{aligned}$$

Nous obtenons bien les résultats escomptés. Il ne reste plus qu'à obtenir x_1 et x_2 . Pour cela nous nous servons de l'égalité des coûts de chaque extrémité de la chaîne $C_1 = C_m$ en fonction de la parité de m . En effet, les extrémités de la chaîne n'ayant qu'un seul sommet voisin contrairement au reste des sommets de la chaîne qui sont de degré 2, leur coût est différent.

Si m est pair.

Soit $m = 2q$, et donc $m - 1 = 2q - 1 = 2(q - 1) + 1$. D'après la formule (3.6) nous avons

$$\begin{cases} x_{m-1} = x_1 - (q-1)x_2 \\ x_m = qx_m \end{cases} \quad (3.7)$$

De $C_1 = C_m$ nous obtenons alors

$$\begin{aligned}
 C_1 = C_m &\Leftrightarrow x_1 + \frac{1}{2}x_2 = x_n + \frac{1}{2}x_{m-1} \\
 &\Leftrightarrow 2x_1 + x_2 = 2qx_2 + x_1 - (q-1)x_2 \\
 &\Leftrightarrow x_1 = qx_2
 \end{aligned}$$

Comme la somme des charges vaut 1 nous pouvons maintenant déduire la valeur de x_2 .

$$\begin{aligned}
 \sum_{i=1}^m x_i = 1 &\Rightarrow x_1 + x_2 + \dots + x_m = 1 \\
 &\Rightarrow qx_2 + \cancel{x_2} + qx_2 - \cancel{x_2} + \cancel{2x_2} + qx_2 - \cancel{2x_2} + \dots \\
 &\quad + qx_2 - \cancel{(q-1)x_2} + qx_2 = 1 \\
 &\Rightarrow \frac{m}{2}qx_2 + qx_2 = 1 \\
 &\Rightarrow x_2 = \frac{1}{(\frac{m}{2} + 1)q}
 \end{aligned}$$

Si m est impair.

Soit $m = 2q + 1$, et donc $m - 1 = 2q$. D'après la formule (3.6) nous avons

$$\begin{cases} x_{m-1} = qx_2 \\ x_m = x_1 - qx_2 \end{cases} \quad (3.8)$$

De $C_1 = C_m$ nous obtenons alors

$$\begin{aligned} C_1 = C_m &\Rightarrow x_1 + \frac{1}{2}x_2 = x_m + \frac{1}{2}x_{m-1} \\ &\Rightarrow 2x_1 + x_2 = 2(x_1 - qx_2) + qx_2 \\ &\Rightarrow x_2 = 0 \end{aligned}$$

Comme la somme des charges vaut 1 nous pouvons maintenant déduire la valeur de x_1 .

$$\begin{aligned} \sum_{i=1}^m x_i = 1 &\Rightarrow x_1 + x_3 + x_5 + \dots + x_m = 1 \\ &\Rightarrow \lceil \frac{m}{2} \rceil x_1 = 1 \\ &\Rightarrow x_1 = \frac{1}{\lceil \frac{m}{2} \rceil} \end{aligned}$$

Ce qui nous donne bien les deux configurations énoncées en fonction de la parité de la longueur de la chaîne. La dernière forme d'équilibre découlant du lemme 3.6 sur les graphes quelconques nous ne la détaillerons pas ici. $\square$

La forme des équilibres sur une chaîne, où deux sommets voisins ne peuvent être non chargés, nous a amenés à considérer cette propriété dans un graphe quelconque.

3.3.1.2 Équilibres atomiques dans un graphe quelconque

En étudiant les cas de graphes simples comme la chaîne, nous avons pu remarquer deux types d'équilibres :

- soit l'ensemble des sommets du graphe est chargé, de telle sorte que chaque sommet ressente le même coût ;
- soit l'ensemble des sommets chargés forment un stable, où tout sommet chargé ressent le même coût et tous les sommets non chargés ont un coût supérieur.

Soit un graphe quelconque $G = (V, E)$ de m sommets, tel que chaque arête entre deux sommets i et j a un poids $d_{i,j} = \frac{1}{2}$, et $d_{i,i} = 1$. Le lemme suivant caractérise la structure d'un équilibre dans un jeu atomique associé à ce graphe.



Lemme 3.6 – Équilibres atomiques dans un graphe quelconque
 $(\forall (i, j) \in E, d_{i,j} = \frac{1}{2})$

Soit un graphe quelconque $G = (V, E)$ de m sommets, avec $m > 3$, tel que chaque arête entre deux sommets i et j a un poids $d_{i,j} = \frac{1}{2}$ et $d_{i,i} = 1$. Il existe un équilibre de Nash où l'ensemble des sommets chargés forment un stable tel que chaque sommet non chargé est voisin d'au moins deux sommets du stable.

Preuve : Nous considérons les trois cas suivants, étant donné un équilibre dont St , l'ensemble des sommets chargés, forme un stable.

- **Cas 1 :** il existe un sommet non chargé n'ayant aucun voisin chargé.
- **Cas 2 :** il existe un sommet non chargé ayant un seul voisin chargé.
- **Cas 3 :** tout sommet non chargé a au moins deux voisins chargés.

Cas 1. Soit S_0 un équilibre dont St , l'ensemble des sommets chargés, forme un stable. Supposons qu'il existe un sommet v non chargé n'ayant aucun voisin chargé. Rappelons que nous avons plus de joueurs que de sommets. Comme $n > m$ dans ce cas il existe forcément un sommet u de St sur lequel sont placés au moins deux joueurs. Un joueur ℓ placé sur u (soit $s_\ell(S_0) = u$) ressent alors un coût $C^\ell = C_u(S_0) \geq 2$. Le sommet v est alors attractif pour ce joueur ℓ . En effet, soit S'_0 la configuration issue de S_0 telle que le joueur ℓ migre en v (soit $s_\ell(S'_0) = v$), alors $C^\ell(S'_0) = C_v(S'_0) = 1$ et donc

$$C^\ell(S'_0) < C^\ell(S_0)$$

Ceci mène donc à une contradiction, S_0 n'est pas un équilibre. Un sommet non chargé a alors au moins un voisin chargé.

Cas 2. Soit S_1 un équilibre dont St l'ensemble des sommets chargés forment un stable. Supposons qu'il existe un sommet v non chargé ayant un voisin chargé u . De la même manière que pour le cas 1, le sommet v est attractif pour un joueur ℓ se trouvant sur u . En effet $C_v(S_1) = 0$ et $C_u(S_1) \geq 2$. Soit S'_1 la configuration issue de S_1 telle que le joueur ℓ migre de u à v , alors

$$C^\ell(S'_1) = 1 + \frac{1}{2}(C_u(S_1) - 1) \leq C_u(S_1)$$

La configuration S_1 n'est donc pas un équilibre, il y a contradiction. Un sommet non chargé a donc au moins deux voisins chargés.

Cas 3. Soit S_2 un équilibre dont St , l'ensemble des sommets chargés, forme un stable. Supposons que tout sommet v non chargé a au moins deux voisins chargés. Notons C^{St} le coût moyen des sommets chargés. De ce fait, chaque sommet non chargé ressent alors au moins $2 \cdot \frac{1}{2}C^{St}$, ce qui le rend inattractif. S_2 est alors bien un équilibre. □

Si tout sommet non chargé a au moins deux voisins dans St , l'ensemble des sommets chargés formant un stable, alors la configuration est un équilibre. Nous avons donc un équilibre où tous les sommets chargés forment un stable tel que chaque sommet non chargé est voisin d'au moins 2 sommets de ce stable. Cette preuve peut être étendue pour toute valeur k telle qu'on ait un graphe $G = (V, E)$ de m sommets, où chaque arête entre deux sommets i et j a un poids $d_{i,j} = \frac{1}{k}$. Pour ce type de graphe, nous aurons un équilibre où tous les sommets chargés forment un stable tel que chaque sommet non chargé est voisin d'au moins k sommets de ce stable.

La complexité de calcul d'équilibres dans le jeu de placement symétrique sans restriction sur la pondération des arêtes ne nous a pas permis d'exhiber les propriétés structurelles de ces équilibres. Néanmoins, nous avons pu le faire pour un cas particulier de pondération. Cette étude de cas nous a mis sur la piste des équilibres dans le cas où les arêtes du graphe ne sont pas pondérées. Dans ce qui suit, nous nous intéressons alors aux équilibres dans le jeu symétrique non pondéré.

3.3.2 Équilibres dans le jeu symétrique non pondéré atomique

Nous avons montré précédemment que le jeu symétrique est un jeu de potentiel. N'ayant pu caractériser la structure des équilibres d'un tel jeu de manière générale, comme nous l'avons fait sur la chaîne, nous avons rajouté une hypothèse supplémentaire concernant les poids des arêtes de notre graphe. Les preuves suivantes considèrent le cas atomique du jeu de placement symétrique.

Nous pouvons montrer qu'étant donné un graphe non orienté $G = (V, E)$ non pondéré (c.-à-d. chaque arête de E entre deux sommets i et j a un poids $d_{i,j} = 1$) nous obtenons un jeu qui est un jeu de potentiel. En effet, le théorème 1.1 étant valable pour toute pondération d'un jeu de placement symétrique, la version non pondérée d'un jeu de placement symétrique est un jeu de potentiel.

De par sa caractéristique de jeu de potentiel, ce jeu possède des équilibres de Nash purs. Nous montrons que ces équilibres sont calculables en temps polynomial, ceux-ci ayant pour forme un stable maximal. Nous prouvons par ailleurs que le meilleur équilibre est le stable maximum du graphe.

Rappelons la définition jeu symétrique non pondéré.



Jeu non pondéré symétrique (ou jeu dans un graphe)

Soient un ensemble de joueurs $N = \{1, \dots, n\}$ tel que chaque joueur $\ell \in N$ a un poids $p_\ell = \frac{1}{n}$ et un graphe non orienté $G = (V, E)$ de m sommets. Chaque arête de E entre deux sommets i et j a un poids $d_{i,j} = 1$ correspondant à l'impact de la charge du sommet i sur le coût de j (et inversement).

Le jeu symétrique non pondéré, ou jeu dans un graphe, est le jeu dans lequel chaque joueur ℓ , ayant pour stratégie $s_\ell = i$, cherche à minimiser

$$C^\ell(x_1, \dots, x_m) = \sum_{j \in N[i]} d_{i,j} x_j(S) = \sum_{j \in N[i]} x_j(S),$$

Soient un équilibre tel que les sommets chargés forment le stable maximum, noté St_{max} et l'ensemble de joueurs $N = \{1, \dots, n\}$ tel que chaque joueur $\ell \in N$ a un poids $\frac{1}{n}$.

Nous souhaitons prouver que pour un graphe $G = (V, E)$ quelconque, la répartition offrant le meilleur coût moyen est un équilibre dont les sommets chargés forment le stable de G de plus grande cardinalité.

Dans un premier temps, nous prouvons le lemme 3.7 qui nous assure que la configuration de coût moyen minimum correspond à un équilibre de Nash. Par la suite, nous montrons le lemme 3.8 nous indiquant que chaque stable maximal de G est un équilibre du jeu et que, de ce fait, le stable maximum est celui de meilleur coût moyen parmi les équilibres sous forme de stable. Enfin, nous montrons le théorème 3.7 qui fait le lien entre l'équilibre correspondant au coût moyen minimum et le stable maximum de G .

Soit S une configuration. Nous définissons le coût moyen de S par

$$C_{moyen}(S) = \sum_{v \in V} x_v(S) \cdot C_v(S),$$

avec $x_v(S) = \sum_{\ell \in N, s_\ell = v} p_\ell$ correspondant à la charge du sommet v . Nous montrons, dans le lemme 3.7, qu'une configuration dont le coût moyen est minimum est un équilibre.



Lemme 3.7 – La configuration de coût moyen minimum est un équilibre de Nash

Soient un graphe non orienté $G = (V, E)$ de m sommets et un ensemble de joueurs $N = \{1, \dots, n\}$, tel que chaque joueur $\ell \in N$ a un poids $p_\ell = \frac{1}{n}$.

Soit S une configuration de coût moyen $C_{moyen}(S) = \sum_{v \in V} x_v(S) \cdot C_v(S)$ minimum. La configuration S est un équilibre de Nash atomique.

Preuve : Comme chaque joueur $\ell \in N$ a un poids $\frac{1}{n}$, alors pour toute configuration S , nous avons par définition

$$\sum_{v \in V} x_v(S) = n \cdot \frac{1}{n} = 1.$$

Cela implique que le coût moyen est

$$\begin{aligned} C_{\text{moyen}} &= \sum_{v \in V} x_v \cdot C_v \\ &= \sum_{v \in V} x_v \sum_{w \in N[v]} x_w \\ &= 2 \sum_{\substack{(v,w) \in E \\ v \neq w}} x_v \cdot x_w + \sum_{v \in V} x_v^2 \end{aligned}$$

Nous cherchons à minimiser la fonction $f(x) = \sum_{\substack{(v,w) \in E \\ v \neq w}} x_v \cdot x_w + \frac{1}{2} \sum_{v \in V} x_v^2$ sous

contraintes que

$$\begin{cases} \sum_{v \in V} x_v = 1 \\ \forall v \in V, x_v \geq 0 \end{cases}$$

Il est clair qu'il existe un vecteur x satisfaisant ces contraintes. Nous associons chacune de ces contraintes à deux fonctions, g et h , dont nous calculons le gradient dans le but de calculer le gradient de la fonction f que nous cherchons à minimiser.

Notons $g(x) = \sum_{v \in V} x_v = 1$ la fonction associée à notre première contrainte. En remarquant que $\frac{\delta g(x)}{\delta x_v} = 1$ nous obtenons que son gradient est

$$\nabla g(x) = \mathbb{1} = (1, \dots, 1)$$

Notons maintenant $\forall v \in V, h_v(x) = x_v$. Le gradient de $h_v(x)$ est le vecteur unitaire $e_v = (0, \dots, 0, \underbrace{1}_v, 0, \dots, 0)$.

Nous pouvons maintenant réécrire le problème de minimisation de la fonction f sous la forme $f(x) = \sum_{\substack{(v,w) \in E \\ v \neq w}} x_v \cdot x_w$ sous contraintes que

$$\begin{cases} g(x) = 1 \\ \forall v \in V, h_v(x) \geq 0 \end{cases}$$

Remarquons qu'il existe x satisfaisant ces contraintes. Soit x^* le vecteur correspondant au minimum de cette fonction f . En appliquant les conditions KKT (Karush-Kuhn-Tucker [Kuhn et Tucker, 1951]) nous pouvons en déduire la condition suivante

$$\nabla f(x^*) = \lambda \nabla g(x^*) + \sum_{v \in V} \mu_v \nabla h_v(x^*)$$

avec λ un réel et $\forall v \in V$, μ_v est un réel positif ou nul tel que si $h_v(x^*) \geq 0$ alors $\mu_v = 0$.

Soit J l'ensemble des sommets non chargés pour x^* , c'est-à-dire $J = \{j | x_j^* = 0\}$. Dans ce cas pour un sommet chargé $v \notin J$ alors $x_v^* \neq 0$ et $h_v(x^*) \geq 0$ donc $\mu_v = 0$. Remarquons que pour tout sommet v

$$\frac{\delta f(x^*)}{\delta x_v} = \sum_{w \in N[v]} x_w^*$$

Il existe alors μ_j et λ tels que

$$\nabla f(x^*) = \sum_{j \in J} \underbrace{\mu_j \cdot e_j}_{>0} + \lambda \mathbb{1}$$

En réécrivant la formule du gradient, nous obtenons alors

$$\nabla f(x^*) = \begin{cases} \left. \begin{matrix} \lambda + \mu_1 \\ \dots \\ \lambda + \mu_k \end{matrix} \right\} |J| = \text{les sommets non chargés} \\ \left. \begin{matrix} \lambda \\ \dots \\ \lambda \end{matrix} \right\} |V - J| = \text{les sommets chargés} \end{cases}$$

Cela nous montre que tous les sommets chargés ont une charge λ identique, tandis que les sommets non chargés ont une charge supérieure à ceux-ci, en effet pour un sommet non chargé $j \in J$ alors $\lambda + \mu_j > \lambda$.

Cette configuration correspond bien à la définition d'un équilibre. Le coût moyen minimum est donc atteint par une configuration qui est un équilibre. $\square$

Le coût moyen minimum est atteint pour une configuration correspondant à un équilibre. Nous montrons maintenant que parmi les équilibres de Nash dont l'ensemble des sommets chargés forme un stable maximal, le stable de plus grande cardinalité est celui de meilleur coût moyen.


Lemme 3.8 – Équilibres dans le jeu symétrique non pondéré

Soient le jeu de placement symétrique non pondéré associé au graphe non orienté $G = (V, E)$, un ensemble de joueurs $N = \{1, \dots, n\}$, tel que chaque joueur $\ell \in N$ a un poids $p_\ell = \frac{1}{n}$, et St_{max} le stable maximum de ce graphe.

Parmi les équilibres de Nash dont l'ensemble des sommets chargés forment un stable maximal, le stable de plus grande cardinalité est celui de meilleur coût moyen, soit $C^{St_{max}} = \frac{1}{|St_{max}|}$.

Preuve : Soit une configuration S dont tous les sommets chargés forment un stable. Par la suite, l'ensemble stable formé par ces sommets est noté St . Nous allons prouver que St est bien un équilibre d'un jeu de placement symétrique non pondéré.

Étant donné S , pour que cette configuration soit un équilibre les coûts de chaque joueur doivent être égaux. Comme pour tout joueur ℓ , ayant pour stratégie le sommet s_ℓ , $C^\ell(S) = C_{s_\ell}(S) = \sum_{v \in N[s_\ell]} x_v(S)$, et que tous les sommets chargés forment un stable, alors

$$\forall \ell \in N, C^\ell(S) = C_{s_\ell}(S) = x_{s_\ell}(S),$$

puisque tout voisin de s_ℓ ne peut être chargé sinon S ne formerait pas un stable.

Maintenant, nous allons définir le coût moyen de cet équilibre. Pour tout couple de joueurs ℓ et k , S est un équilibre si et seulement si $C^\ell = C^k$. Comme $\forall \ell \in N, C^\ell(S) = x_{s_\ell}(S)$ alors $\forall \ell, k \in N, C^\ell = C^k \Rightarrow x_{s_\ell}(S) = x_{s_k}(S)$.

Cela implique que pour que S soit un équilibre, tous les sommets chargés doivent avoir la même charge. Pour cela une répartition uniforme des joueurs sur le nombre de sommets du stable est effectuée.

$$\forall v \in St, C^{St} \leq C_v \leq C^{St} + 1$$

$$\text{avec } C^{St} = \frac{\sum_{v \in V} x_v(S)}{|St|} = \frac{1}{|St|}.$$

De ce fait, plus la taille du stable est grande, moindre est le coût et donc le stable de cardinalité maximale est celui de coût le plus faible parmi les stables. Le stable maximum est donc celui de meilleur coût moyen avec

$$C^{St_{max}} = \frac{1}{|St_{max}|}$$

□

Nous avons prouvé précédemment que dans le jeu de placement symétrique non pondéré le coût moyen minimum correspond à un équilibre de Nash. Nous montrons maintenant que ce coût moyen minimum est atteint lorsque l'équilibre correspond au stable maximum du graphe.

Le théorème suivant est dédié à prouver que le stable maximum du graphe non orienté $G = (V, E)$ est le meilleur équilibre en termes de coût pour le jeu de placement non pondéré symétrique.



Théorème 3.7 – Meilleur équilibre dans le jeu symétrique non pondéré

Soient un graphe non orienté $G = (V, E)$ et un ensemble de joueurs $N = \{1, \dots, n\}$. Chaque joueur $\ell \in N$ a un poids $p_\ell = \frac{1}{n}$ et cherche à minimiser

$$C^\ell(S) = \sum_{v \in N[s_\ell]} x_v(S)$$

avec S une configuration et $x_v(S) = \sum_{\substack{\ell \in N \\ s_\ell = v}} p_\ell$.

Le stable maximum du graphe est le coût moyen minimum pour ce jeu.

Preuve : D'après le lemme 3.8, parmi les équilibres qui sont des stables, le stable maximum St_{max} est celui de meilleur coût moyen, celui-ci étant de $C^{St_{max}} = \frac{1}{|St_{max}|}$.

Nous cherchons à savoir si le coût moyen minimum est meilleur que $\frac{1}{|St_{max}|}$, sachant que

$$\begin{aligned} C_{moyen}(S) &= \sum_{v \in V} x_v \cdot C_v \\ &= \sum_{v \in V} x_v \sum_{w \in N[v]} x_w \\ &= 2 \sum_{(v,w) \in E} x_v \cdot x_w + \sum_{v \in V} x_v^2 \end{aligned}$$

D'après Lovasz [Lovasz, 1994], étant donné un graphe $G = (V, E)$ et α_G la plus grande cardinalité d'un stable de G alors

$$1 - \frac{1}{\alpha_G} = \max \left\{ 2 \sum_{(v,w) \notin E} x_v \cdot x_w \mid \forall v \in V, x_v \geq 0, \sum_{v \in V} x_v = 1 \right\} \quad (3.9)$$

Nous pouvons montrer, qu'étant donné un vecteur x satisfaisant la condition suivante $\{\forall v \in V, x_v \geq 0, \sum_{v \in V} x_v = 1\}$, alors

$$\begin{aligned} 1 - \frac{1}{\alpha_G} &= \max\{2 \sum_{(v,w) \notin E} x_v \cdot x_w\} \Leftrightarrow \frac{1}{\alpha_G} = \min\{2 \sum_{(v,w) \in E} x_v \cdot x_w + \sum_{v \in V} x_v^2\} \\ &\Leftrightarrow \frac{1}{\alpha_G} = \min\{C_{moyen}(S)\} \end{aligned}$$

En effet

$$\begin{aligned} 2 \sum_{(v,w) \notin E} x_v \cdot x_w &= \left(\sum_{v \in V} x_v \sum_{w \in V} x_w \right) - 2 \sum_{(v,w) \in E} x_v \cdot x_w - \sum_{v \in V} x_v^2 \\ &= 1 - 2 \sum_{(v,w) \in E} x_v \cdot x_w - \sum_{v \in V} x_v^2 \end{aligned}$$

En remplaçant dans la formule (3.9), le résultat suivant est obtenu

$$\begin{aligned} 1 - \frac{1}{\alpha_G} &= \max\{2 \sum_{(v,w) \notin E} x_v \cdot x_w\} \\ 1 - \frac{1}{\alpha_G} &= \max\{1 - 2 \sum_{(v,w) \in E} x_v \cdot x_w - \sum_{v \in V} x_v^2\} \\ 1 - \frac{1}{\alpha_G} &= 1 + \max\{-2 \sum_{(v,w) \in E} x_v \cdot x_w - \sum_{v \in V} x_v^2\} \\ -\frac{1}{\alpha_G} &= -\min\{2 \sum_{(v,w) \in E} x_v \cdot x_w + \sum_{v \in V} x_v^2\} \\ \frac{1}{\alpha_G} &= \min\{2 \sum_{(v,w) \in E} x_v \cdot x_w + \sum_{v \in V} x_v^2\} \\ \frac{1}{\alpha_G} &= \min\{C_{moyen}(S)\} \end{aligned}$$

Cela implique que le coût moyen minimum est de $\frac{1}{\alpha_G}$. Sachant que α_G est la cardinalité maximale d'un stable de G alors cela équivaut à $\frac{1}{|St_{max}|}$.

Le stable maximum est donc l'équilibre de meilleur coût moyen pour un jeu symétrique non pondéré. $\square$

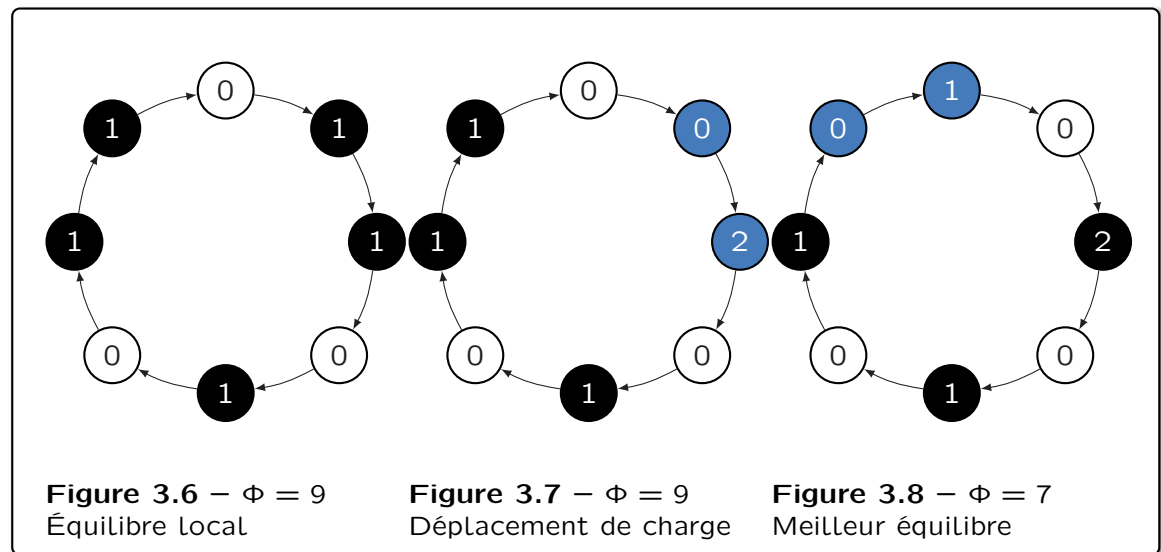
3.4

Conclusion

Dans le chapitre précédent, nous avons défini notre jeu de placement. Nous avons également vu que certaines variantes de ce jeu ne possèdent pas d'équilibre de Nash. Nous nous sommes alors posé la question fondamentale de savoir dans quels cas il existait des équilibres de Nash et si tel est le cas s'ils étaient facilement calculables.

Nous avons montré que déterminer l'existence d'un équilibre de Nash pur du jeu de placement, sans contraintes sur les coefficients des arcs, est NP-Complet, pour les jeux asymétriques. Néanmoins, en faisant l'hypothèse que les coefficients des arcs entre chaque paire de sommets sont égaux, nous avons obtenu un jeu de potentiel. En effet, considérer un tel graphe orienté ayant pour tout couple de sommets (i, j) des coefficients égaux (c.-à-d. $d_{i,j} = d_{j,i}$) revient à étudier le jeu sur un graphe non orienté ce qui nous ramène au jeu de placement symétrique. Nous avons montré que ce type de jeu est bien un jeu de potentiel et nous sommes intéressés à la forme de ses équilibres et à la complexité liée à leur calcul. Nous avons pu caractériser les équilibres dans un cas particulier de jeu de placement symétrique et montré que le cas général est PLS-complet. Cela nous a amenés à considérer une variante non pondérée du jeu symétrique qui est également un jeu de potentiel et pour lequel nous avons prouvé qu'il existe des équilibres sous forme de stable maximal. Nous avons également pu prouver que le stable maximum est le meilleur équilibre pour ce type de jeu.

En complément de ces résultats sur l'existence des équilibres et leur structure, nous pouvons nous poser la question du calcul distribué de tels équilibres. En utilisant la démonstration du théorème 3.2, nous pouvons montrer que lorsque nous nous trouvons sur un équilibre local qui possède au moins deux sommets voisins chargés nous pouvons nous réduire à un équilibre sous forme de stable sans augmenter le coût moyen. En effet, en partant d'une configuration S étant un équilibre de Nash tel que deux sommets voisins 1 et 2 sont chargés, nous pouvons en déduire une configuration S' telle que seules les charges x_1 et x_2 aient été modifiées. Nous pouvons supposer alors, sans perte de généralité, un transfert de charges t du sommet 1 vers 2, avec t tel que les contraintes sur le vecteur x , $\forall v \in V, x_v \geq 0$ et $\sum_{v \in V} x_v = 1$ soient vérifiées, soit $-x_2 \leq t \leq x_1$.



La figure ci-dessus illustre l'amélioration d'un équilibre local sur un cycle non pondéré. Partant du cycle représenté sur la figure 3.6, qui est un équilibre local qui possède au moins deux sommets voisins chargés, nous pouvons nous réduire à un équilibre sous forme de stable, comme sur la figure 3.8, sans augmenter le coût moyen. Nous passons par une étape intermédiaire en déplaçant la totalité de la charge d'un des deux sommets voisins chargés vers le second comme illustré par les sommets colorés en bleu sur la figure 3.7. Cet état obtenu n'est pas un équilibre, mais n'affecte en rien le coût moyen. Une meilleure réponse nous permet d'atteindre un équilibre de meilleur coût tel que sur la figure 3.8.

En complément des résultats sur l'existence et la complexité du calcul des équilibres dans le jeu de placement, nous pouvons nous intéresser à comment calculer ces équilibres de manière distribuée. Le jeu de placement présenté est un jeu de congestion généralisé. Nous nous intéressons alors au calcul distribué des équilibres dans des jeux de potentiel particuliers avec pour perspective d'étendre ces méthodes au jeu de placement. Dans le chapitre suivant, nous nous intéresserons à ce type de mécanismes, permettant le calcul distribué d'équilibres de Nash, avec notamment l'algorithme de meilleure réponse.

Partie II

Techniques d'apprentissage appliquées à des jeux de congestion et de potentiel



Résumé

Dans cette partie nous nous intéressons à l'apprentissage d'équilibres de Nash dans des jeux de potentiel par divers algorithmes d'apprentissage et de minimisation de regret.

Le chapitre 4 présente la notion d'apprentissage. Nous introduisons la dynamique de meilleure réponse, la dynamique de réplcation, ainsi qu'un algorithme de minimisation de regret associé aux problèmes de Multi-Armed Bandit.

Dans le chapitre 5 nous étudions la convergence vers des équilibres dans un jeu basé sur un problème issu de la théorie des graphes : Max-Cut, que nous étendons à des jeux d'ordonnancement pour des applications à la gestion d'interférences dans des cellules mobiles.

Nous nous focalisons dans le chapitre 6 sur un autre système en concurrence, où nous comparons algorithmes d'apprentissage et minimisation de regret.

Techniques d'apprentissage d'équilibres

Dans les chapitres précédents, nous avons introduit l'équilibre de Nash ainsi que les jeux de potentiel, qui ont la particularité de toujours posséder un de ces équilibres. Le chapitre 3 nous a permis de discuter l'existence et la calculabilité des équilibres de Nash dans le jeu de placement. En complément de ces résultats, nous pouvons nous intéresser à comment calculer ces équilibres de manière distribuée.

Dans la deuxième partie de cette thèse, nous nous sommes donc focalisés sur des jeux de potentiel particuliers. Nous nous sommes intéressés au calcul distribué de leurs équilibres, avec pour perspective d'étendre ces méthodes au jeu de placement qui est un cas plus général. Ce chapitre a pour vocation d'établir un état de l'art sommaire sur les algorithmes d'apprentissage et de minimisation de regret. Nous présenterons trois algorithmes que nous utiliserons, dans les chapitres suivants, pour le calcul d'équilibres de Nash dans des jeux de potentiel basé sur un problème de théorie des graphes : Max-Cut.

Nous nous intéressons ici à la construction d'équilibres dans des jeux non coopératifs répétés et à information incomplète. L'information incomplète vient du fait que les différents joueurs n'ont qu'une vision locale du jeu : ils ne connaissent pas les stratégies des autres joueurs ni leur fonction d'utilité. À chaque fois que le jeu est répété, les joueurs doivent choisir une stratégie à jouer. Ils perçoivent alors une récompense, ou gain, dépendant de l'état du système. En fonction de la récompense obtenue, ils réactualisent le choix de leur stratégie à l'aide d'un algorithme afin d'améliorer leur propre récompense. L'impact de chaque joueur sur le système est faible, toutefois, comme le nombre d'acteurs est grand, une évolution globale du système peut se produire.

Nous nous focalisons sur des algorithmes qui réalisent de petits ajustements (stochastiques ou déterministes) à chaque répétition du jeu. Nous supposons ces ajustements complètement répartis puisque tous les joueurs ont une vue locale du système. Nous présentons la dynamique de meilleure réponse où, à chaque tour de jeu, un unique joueur effectue un mouvement appelé meilleure réponse, que nous avons introduit dans le chapitre 1. La dynamique de meilleure réponse est un algorithme simple qui garantit une convergence vers un équilibre de Nash dans des jeux de potentiel [Rosenthal, 1973]. Néanmoins, une telle dynamique n'assure pas de converger rapidement vers un équilibre. De plus, cette

convergence n'est également pas garantie dans le cas où chaque joueur effectue une meilleure réponse de manière simultanée.

Nous intéressons à du calcul distribué d'équilibre, une action simultanée des joueurs doit être envisagée. Nous considérons alors la dynamique de réplcation avec l'algorithme "Linear Reward Inaction" (LRI). Pour des jeux où tous les joueurs ont le même gain, il a été prouvé que l'algorithme LRI [Sastry et al., 1994] converge vers un équilibre de Nash pur néanmoins sans qu'aucun temps de convergence n'ait été calculé.

Beaucoup de problèmes d'optimisation et d'apprentissage peuvent être modélisés sous forme d'un problème de minimisation de regret. Il est possible d'utiliser des algorithmes de minimisation de regret, liés au problème de "Multi-Armed Bandit", pour converger vers un équilibre de Nash. Les algorithmes de Multi-Armed Bandit [Auer et al., 2002, Audibert et Bubeck, 2009] sont utiles pour trouver un équilibre de Nash dans les jeux à somme nulle [Grigoriadis et Khachiyan, 1995, Audibert et Bubeck, 2009], notamment l'algorithme EXP3 [Auer et al., 2002]. Il a été prouvé [Grigoriadis et Khachiyan, 1995] que si les deux joueurs utilisent tous deux l'algorithme EXP3 pour le choix de leur stratégie, alors le système converge vers un équilibre de Nash mixte.

Sommaire

5.1	Coupe maximale dans un graphe non pondéré	93
5.1.1	Recherche d'une coupe maximale locale	97
5.1.2	Coupe localement maximale dans un graphe complet	98
5.1.3	Coupe maximum locale dans des graphes généraux	101
5.1.4	Autres topologies de graphes et simulations	104
5.1.5	Lien avec l'interférence inter cellule	106
5.2	Coordination d'interférences inter cellules	107
5.2.1	Modélisation du système	108
5.2.2	Apprentissage distribué des équilibres de Nash purs	112
5.2.3	Convergence vers un équilibre de Nash pur de l'algorithme	113
5.2.4	Expérimentations	120
5.3	Conclusion	122

4.1**Dynamique de meilleure réponse**

Nous avons défini l'équilibre de Nash comme étant un état dans lequel aucun joueur ne peut, étant données les stratégies jouées par les autres joueurs, choisir une action différente lui permettant d'améliorer son utilité. Comme nous l'avons indiqué précédemment, nous nous plaçons dans le cadre de jeux répétés.

L'objectif de la dynamique de meilleure réponse est d'atteindre cet équilibre par de petits ajustements, en modifiant, à chaque tour, la stratégie d'un unique joueur de manière à améliorer son utilité, tout en fixant les stratégies des autres joueurs. C'est ce changement de stratégie d'un joueur pour une stratégie améliorant son utilité que l'on nomme une *meilleure réponse*, comme défini dans le chapitre 1. Celle-ci peut être simplement une *bonne réponse* ou bien une *meilleure réponse*, c'est-à-dire celle qui améliore au mieux l'utilité du joueur (cf. définitions 1.1 et 1.2).

La dynamique de meilleure réponse consiste donc à choisir un joueur selon une politique fixée préalablement et modifier sa stratégie pour celle maximisant son utilité (ou minimisant son coût) et réitérer ce processus jusqu'à ce qu'aucun joueur ne puisse effectuer de meilleure réponse.

Algorithme 4.1 – Algorithme de meilleure réponse

Initialisation : Chaque joueur i choisit une stratégie pure

Pour chaque itération t faire

 Choisir un joueur i selon la politique fixée

 Choisir la meilleure réponse pour ce joueur i , soit la (ou une s'il en existe plusieurs) stratégie s_i qui minimise/maximise son utilité

Fin

Dans le chapitre 1, nous avons présenté les jeux de potentiel dont une des particularités est qu'ils possèdent toujours au moins un équilibre de Nash pur. La preuve du théorème de Monderer et Shapley [Monderer et Shapley, 1996b] peut se voir comme un algorithme d'apprentissage d'équilibres et plus particulièrement une dynamique de meilleure réponse.

En effet, l'idée de cette preuve est de montrer que pour un jeu de potentiel, il existe une suite finie de profils de stratégies, où un unique joueur effectue une meilleure réponse entre chaque profil, menant à un équilibre de Nash. Pour plus de détails, se reporter à la preuve du théorème 1.1.

Cette preuve de l'existence d'équilibres de Nash purs dans les jeux de potentiel est également une preuve que la dynamique de meilleure réponse converge vers un équilibre de Nash pur : si les joueurs jouent chacun leur tour en améliorant systématiquement leur profit quand ils le peuvent, alors le jeu converge vers un équilibre de Nash pur. Néanmoins, cette convergence n'est garantie que lorsque les joueurs effectuent une meilleure réponse de manière séquentielle. Dans le cas contraire, si les joueurs effectuent à chaque tour de jeu une meilleure réponse de manière simultanée, alors cette preuve de convergence ne tient plus. Nous pouvons également ajouter que, malgré l'assurance de converger vers un équilibre de Nash, cela n'induit pas une convergence rapide vers cet équilibre.

La dynamique de meilleure réponse ne nous donne pas de résultats quant à la convergence vers un équilibre lorsque les joueurs effectuent leurs actions de manière simultanée. Nous nous intéressons alors à une autre dynamique, la dynamique de réplication, dont la convergence a été étudiée pour des jeux où les joueurs agissent simultanément.

4.2

Dynamique de réplication

Certains algorithmes, comme "*Linear Reward Inaction*" (*LRI*) [Sastry *et al.*, 1994] permettent de converger vers une situation stable qui est un équilibre de Nash pur. L'algorithme *LRI* est un algorithme d'apprentissage par renforcement décentralisé qui s'appuie sur des informations locales uniquement, ce qui en fait un bon candidat pour l'apprentissage d'équilibres dans des jeux à informations incomplètes.

L'algorithme *LRI* utilise un vecteur de probabilité pour la sélection de la stratégie d'un joueur. Notons que chaque élément de ce vecteur indique la probabilité que la stratégie correspondante soit choisie. Initialement identiques, ces probabilités sont mises à jour à chaque tour du jeu. Selon la récompense obtenue à chaque tour, le vecteur de probabilité est mis à jour en fonction d'un facteur $b \in [0, 1]$. En fonction de la valeur de b , l'algorithme converge plus ou moins rapidement vers une solution.

Cet algorithme, appliqué sur plusieurs itérations du même jeu, permet aux joueurs d'apprendre la meilleure stratégie à choisir malgré les préférences et les stratégies inconnues de leurs opposants. L'algorithme est itéré jusqu'à sa convergence. À chaque itération (ou tour) t , l'algorithme d'apprentissage exécute les trois actions suivantes pour chacun des joueurs :

- Choisir une stratégie en fonction du vecteur de probabilités des stratégies ;

- Obtenir une récompense qui dépend du choix de sa stratégie ;
- Mettre à jour le vecteur de probabilités des stratégies suivant une règle qui dépend de l'algorithme d'apprentissage utilisé.

4.2.1 Notations et algorithme *LRI*

Dans l'algorithme 4.2, nous présentons l'algorithme *LRI*. Dans cet algorithme, à chaque tour, tous les joueurs choisissent leur action en parallèle. Une fois que tous les joueurs ont effectué leur action, ils perçoivent leur récompense et mettent à jour leur vecteur de probabilité.

Chaque joueur ℓ cherche à apprendre la meilleure stratégie parmi ses K stratégies. Étant donné un joueur ℓ , nous notons $P_\ell(t)$ le vecteur de probabilité de celui-ci au temps t et donc $P_{s,\ell}(t)$ la probabilité qu'a la stratégie s d'être sélectionnée au temps t . Soit $S(t) = (s_1(t), \dots, s_\ell(t), \dots)$ le profil de stratégie où $s_\ell(t)$ est la stratégie pure choisie au temps t par le joueur ℓ . Pour chaque joueur ℓ , le vecteur de probabilité est initialisé de la manière suivante : $\forall s \in S_\ell, P_{s,\ell}(0) = \frac{1}{K}$, avec K le nombre de stratégies.

L'exécution de l'algorithme *LRI* requiert des récompenses comprises entre 0 et 1. Pour cela, nous notons alors $u_\ell(S(t))$ un réel entre 0 et 1 représentant l'utilité du joueur ℓ à l'itération t en fonction de $S(t)$.

Algorithme 4.2 – Algorithme d'apprentissage *LRI*

Données : $b \in [0, 1]$ un réel

Initialisation : $\forall \ell \in N, s \in \{1, \dots, K\}, P_{s,\ell}(1) = \frac{1}{K}$

Pour chaque itération t faire

Pour chaque joueur ℓ faire

 Tirer une stratégie s aléatoirement en respectant le vecteur de probabilités $P_\ell(t)$

Fin

Pour chaque joueur ℓ faire

 Recevoir un gain $u_\ell(S(t)) \in [0, 1]$

 Mettre à jour les probabilités des stratégies selon la règle (4.1)

Fin

Fin

Pour un joueur ℓ et une stratégie s , la formule (4.1) indique la règle de mise à jour du vecteur P_ℓ en fonction de $S(t)$.


Règle de mise à jour du vecteur de probabilités de LRI

$$P_{s,\ell}(t+1) = \begin{cases} P_{s,\ell}(t) - b \cdot u_\ell(S(t)) \cdot P_{s,\ell}(t) & \text{si } s \neq s_\ell(t) \\ P_{s,\ell}(t) + b \cdot u_\ell(S(t)) \cdot (1 - P_{s,\ell}(t)) & \text{si } s = s_\ell(t) \end{cases} \quad (4.1)$$

4.2.2 Propriétés de l'algorithme

Dans ce qui suit, nous listons les propriétés de l'algorithme LRI selon le jeu considéré. Pour un jeu de deux joueurs à somme nulle, l'algorithme LRI converge vers un équilibre de Nash pur s'il en existe. Pour les jeux à n joueurs et à intérêt commun, l'algorithme LRI converge vers un équilibre de Nash pur. La dynamique de cet algorithme pour des jeux à somme générale a été étudiée par Sastry *et al.* [Sastry *et al.*, 1994].


Théorème 4.1 – Convergence de LRI [Sastry *et al.*, 1994]

Considérons la suite des processus interpolés $\{P^b(\cdot)\}$.

Soit $X_0 = P^b(0) = P(0)$. Alors la séquence converge faiblement, lorsque b tend vers 0, vers $X(\cdot)$ qui est la solution de l'équation différentielle ordinaire

$$\frac{dX}{dt} = f(X), \quad X(0) = X_0$$

Pour la dynamique de l'algorithme LRI , les propriétés suivantes caractérisent les solutions de l'équation différentielle ordinaire et donc le comportement à long terme de l'algorithme [Sastry *et al.*, 1994] :

- tous les équilibres de Nash sont des points stationnaires ;
- tous les équilibres de Nash strict sont asymptotiquement stables ;
- tous les points stationnaires qui ne sont pas des équilibres de Nash purs ne sont pas stables.

L'intérêt de la méthode LRI est de converger vers un équilibre de Nash pour des jeux à plusieurs joueurs cherchant à optimiser une même fonction d'utilité, et ce malgré une action simultanée des joueurs. Cette approche est donc préférable à la dynamique de meilleure réponse pour le calcul distribué d'équilibres, auxquels nous nous intéressons par la suite. Néanmoins, comme pour la dynamique de meilleure réponse, la convergence vers un équilibre peut être longue. En effet, la règle de mise à jour de l'algorithme LRI , présentée dans la formule (4.1) où $0 < b < 1$, a la propriété suivante : la probabilité d'une stratégie choisie ne peut qu'augmenter ou se stabiliser à chaque itération, les utilités étant positives ou nulles. Nous arrivons alors à une situation d'équilibre lorsque les vecteurs de probabilité de chaque joueur

ont une stratégie ayant pour probabilité 1. Cependant, pour chaque joueur, plus la probabilité d'une stratégie se rapproche de la valeur 1 et moins vite elle augmentera. La vitesse de convergence est modulée par le paramètre b . Ceci permet d'éviter que le vecteur de probabilités des stratégies d'un joueur ne converge très vite vers une stratégie et s'y fige. Cette tendance à se figer sur une stratégie peut-être problématique dans un contexte très dynamique, comme nous le verrons dans le chapitre 6. En effet, quand le vecteur de stratégies d'un joueur se fige, ce dernier n'a plus la possibilité de changer de stratégie. Cependant, cette stratégie peut être optimale au moment choisi, mais, si les vecteurs de probabilité des autres joueurs sont loin d'avoir convergé, cette stratégie peut ne plus être idéale pour ce joueur.

4.3

Minimisation de regret et problème de bandits manchots

4.3.1 Regrets et minimisation

La prise de décisions répétitives dans des environnements incertains comme c'est le cas dans les problèmes de routage ou de trafic routier, est appelée apprentissage "online". Le modèle d'apprentissage "online" considéré est le suivant : pendant un nombre de tours fixé à l'avance, à chaque tour de jeu, un preneur de décision choisit une action parmi un (éventuellement continu) ensemble d'actions et gagne une récompense (ou subit une perte) déterminée par une fonction dépendant du temps et de l'action choisie. Les informations sur la valeur de cette récompense ne sont révélées au joueur qu'après qu'il ait effectué son action. L'objectif du joueur est de maximiser sa récompense (ou, respectivement, de minimiser son coût) à long terme de façon adaptative. Par la suite, nous parlerons uniquement de problèmes où l'objectif du joueur est de minimiser sa perte.

La notion de regret consiste à comparer les performances de l'algorithme "online" choisi avec celles d'une politique alternative, pour les T tours effectués, de manière rétrospective. Le regret représente la différence entre la perte cumulée subie en exécutant l'algorithme "online" et la perte cumulée qui aurait été obtenue avec la politique alternative.

Il existe plusieurs notions de regrets, le *regret externe*, le *regret interne* ou le *regret swap*, qui quantifient de manière différente la perte entre l'algorithme utilisé et la politique alternative. Les problèmes de "*Multi-Armed Bandit*" stochastique qui nous intéressent, et que nous présentons par la suite, sont des jeux répétés où un joueur doit choisir une action, à chaque tour de jeu, pendant une période fixée dans le but de minimiser sa perte cumulée. Ces problèmes

sont connus pour posséder des méthodes de minimisation du regret externe optimales telles que l'algorithme *UCB* [Auer et al., 2002]. Dans la littérature, certains algorithmes minimisant le regret ont été considérés pour calculer des équilibres. Dans un jeu qui minimise le regret interne, la distribution empirique du jeu est connue pour converger vers un équilibre corrélé. Pour plusieurs jeux classiques, plusieurs travaux montrent que si tous les participants minimisent leur propre regret externe, le trafic global est garanti de converger vers un équilibre de Nash approximatif. Nous nous intéressons, dans le chapitre 6, à l'application de la méthode *UCB*, de manière distribuée, dans le cadre de jeux à plusieurs joueurs afin de comparer l'algorithme de minimisation de regret avec le calcul distribué d'équilibre par la méthode *LRI*.

L'objectif, pour minimiser le regret externe, est de concevoir un algorithme "online" qui sera en mesure d'approcher les performances de l'algorithme offrant la meilleure perte cumulée. Le regret externe compare les performances obtenues à la meilleure action en rétrospective. La politique alternative consiste à sélectionner toujours la même action : celle donnant la perte cumulée la plus faible rétrospectivement.

Notons H l'algorithme "online" utilisé pour T tours et l_H^t la perte subie au temps t pour cet algorithme. La perte cumulée de l'algorithme les T tours est alors

$$L_H^T = \sum_{t=1}^T l_H^t$$

Soit $L_i^T = \sum_{t=1}^T l_i^t$ la perte cumulée de l'action i pour les T tours.

Nous notons alors $L_{min}^T = \min_i L_i^T$ la perte cumulée de l'action offrant la perte cumulée la plus faible sur les T tours.

Nous pouvons alors définir le regret externe comme suit.



Définition 4.1 – Regret externe pour T tours de jeu

Le regret externe pour un algorithme H pour T tours est le suivant

$$R = L_H^T - L_{min}^T$$

Pour le regret interne et swap, la politique alternative considère la séquence complète de l'algorithme "online" et y effectue une modification simple. Cette modification peut consister à remplacer une unique action de la séquence, par exemple à chaque fois que l'action A apparaît dans la séquence de l'algorithme "online", elle est remplacée par l'action B dans la politique alternative, tandis que toutes les autres actions restent identiques à celle de l'algorithme "online". Le regret interne est obtenu en comparant la séquence d'actions de l'algorithme "online" avec une séquence alternative où une et une seule action de la

séquence d'origine est remplacée à chaque occurrence.

Le regret swap s'obtient en remplaçant un ensemble d'actions de la séquence de l'algorithme "online" par un autre.

Par la suite, nous nous intéressons aux problèmes de Multi-Armed Bandit (ou Bandits Manchots) qui font partie de ces modèles d'apprentissage "online" afin de minimiser le regret externe.

4.3.2 Problème de Multi-Armed Bandit (ou Bandits Manchots)

L'étude des problèmes de *Multi-Armed Bandit (MAB)* est cruciale dans l'analyse de la prise de décisions en milieu incertain. Ce problème a été un secteur de recherche actif depuis 1950. Le problème a été cité comme suit [Gittins et Gittins, 1979] : il y a K leviers, chacun a une probabilité de succès inconnue pour émettre un gain. Les probabilités des succès des leviers sont supposées indépendantes. L'objectif est de tirer les leviers de façon séquentielle de manière à maximiser une récompense totale sur une période de temps T fixée à l'avance et connue du joueur. Plusieurs politiques ont été proposées pour ce problème [Lai et Robbins, 1985, Grigoriadis et Khachiyan, 1995, Auer *et al.*, 2002, Audibert et Bubeck, 2009].

Le problème de Multi-Armed Bandit consiste à décider quel levier tirer afin de maximiser une récompense totale pour une série d'essais. Les K leviers représentent généralement les bras de K machines à sous, d'où l'appellation de bandits manchots. La loi des récompenses étant inconnue du joueur, celui-ci est confronté au compromis fondamental entre accumuler de l'expérience afin d'obtenir de l'information sur les récompenses reçues et jouer la machine qui semble la plus prometteuse. Le problème des bandits manchots réside sur le dilemme entre *exploration* et *exploitation*, appelé dilemme exploration-exploitation.

Le dilemme exploration-exploitation est un défi important pour l'apprentissage par renforcement. En effet, le joueur ne connaît pas la distribution de probabilité des machines.

Afin de trouver la machine ayant une espérance de gain maximale, le joueur doit estimer l'espérance de gain des différentes machines. Pour cela, il doit explorer les récompenses des différentes machines pour améliorer son estimation. Quand le joueur commence à accumuler une certaine connaissance sur l'environnement, il aura le choix entre exploiter la machine qu'il pense être optimale pour augmenter ses gains ou explorer d'autres machines, dans le but de découvrir d'autres machines optimales.

Si le joueur joue la machine qui lui semble meilleure (exploitation), il peut passer à côté d'une autre machine qui a une espérance de gain plus grande. L'exploitation d'une machine dont l'espérance de gain semble la plus intéressante à un moment donné peut pourtant s'avérer un mauvais choix, son espérance pouvant décroître avec le temps le choix d'une autre machine serait

alors nécessaire. Cependant, s'il met beaucoup de temps à essayer toutes les machines (exploration), il perd l'occasion de jouer plusieurs fois la meilleure machine pour avoir un meilleur gain. Il effectue donc une exploration au détriment de l'exploitation. En effet une phase d'exploration peut induire des gains inférieurs à une phase d'exploitation de la meilleure machine courante et donc une perte sur l'espérance des gains du joueur. Un compromis entre exploitation et exploration est donc nécessaire à la maximisation de l'espérance des gains du joueur.

Dans ce qui suit, nous formulons le problème de Multi-Armed Bandit.

Formalisation du problème de Multi-Armed Bandit

Dans le problème de Multi-Armed Bandit, un joueur fait face à K machines de jeu indépendantes pour un temps fixé T . À chaque tour de jeu, le joueur choisit une action, et reçoit une récompense de la machine associée. Chaque machine a une distribution de probabilité stationnaire inconnue du joueur pour récompense.

Dans ce cas précis, l'adversaire est considéré comme stochastique. Chaque machine i a une distribution de probabilité P_i ainsi qu'une espérance μ_i . Au tour $t = 1, \dots, T$, le joueur choisit une machine parmi l'ensemble des K machines d'après les informations du passé (actions et récompenses observées).

À la fin des T tours, le regret externe du joueur est évalué. Pour rappel, le regret externe évalue la perte entre la séquence d'actions choisies par le joueur et la séquence qui, à chaque tour, choisit la machine dont la récompense est optimale pour les T tours. Précisément, cette perte est due au fait que, à chaque tour, la politique ne sélectionne pas toujours la machine optimale.

D'après la définition 4.1 du regret externe, nous pouvons définir le regret externe après T tours sur K machines.



Définition 4.2 – Regret externe après T tours sur K machines (adversaire stochastique)

$$R_T = \mu^* \times T - \sum_{j=1}^K \mu_j \times T_j \quad (4.2)$$

où μ^* est l'espérance de la machine optimale, μ_j l'espérance de la machine j et T_j le nombre de fois où la machine j a été jouée.

Le regret d'une politique fixée est la perte attendue après un nombre fixé de tours T . Plus la phase d'exploration est importante, plus le regret du joueur augmente. En outre, une mauvaise estimation et donc le choix d'une mauvaise machine à exploiter conduisent à une augmentation du regret.

Beaucoup de problèmes d'optimisation et d'apprentissage peuvent être modélisés sous forme d'un problème de Multi-Armed Bandit. Différents algorithmes ont été proposés comme solution au problème de Multi-Armed Bandit pendant

les deux dernières décennies. Dans ce qui suit, nous présentons un algorithme de minimisation de regret utilisé dans le cadre du problème des Multi-Armed Bandit stochastique à K bras.

4.3.3 Algorithme de minimisation de regret : *UCB*

La méthode appelée "*Upper Confidence Bound*" (*UCB*) [Auer et al., 2002] est utilisée dans le cadre du problème de Multi-Armed Bandit stochastique.

Le principe de *UCB* réside dans le calcul d'un indice pour chaque machine. À chaque itération, la machine dont l'indice est maximum, parmi l'ensemble des indices des machines, est sélectionnée. Le joueur perçoit alors une récompense selon la machine sélectionnée. Cet indice est ensuite mis à jour pour chacune des machines.

Nous considérons K machines. Notons $indice_j(t)$ l'indice de la machine j au temps, calculé par $indice_j(t) = \bar{x}_j(t) + \sqrt{\frac{2 \ln t}{T_j}}$, avec T_j , le nombre de fois que la machine j a été sélectionnée. Cet indice se compose de deux éléments, que l'on nomme facteur d'exploitation et facteur d'exploration. Le facteur d'exploitation correspond simplement à la récompense moyenne de la machine calculée sur t la période de temps écoulé, noté $\bar{x}_j(t)$ pour la machine j . Quant au facteur d'exploration $\sqrt{\frac{2 \ln t}{T_j}}$ il dépend de T_j , le nombre de fois que la machine j a été sélectionnée, ainsi que du nombre total d'itérations effectuées t . De ce fait lorsqu'une machine n'a pas souvent été sélectionnée, car sa récompense moyenne $\bar{x}_j$ est trop faible, le facteur d'exploration dépendant de T_j , le nombre de sélections de la machine, augmente et permet alors de réévaluer une machine laissée de côté.

L'algorithme 4.3 présente l'algorithme *UCB* pour un joueur affrontant K machines.

Algorithme 4.3 – Algorithme *UCB* [Auer et al., 2002]

Données : K le nombre de machines.

Initialisation : Explorer chaque machine j et fixer $\bar{x}_j$

Pour t de 1 à T faire

Pour chaque machine j faire

 Calculer $indice_j(t) = \bar{x}_j(t) + \sqrt{\frac{2 \ln t}{T_j}}$ en fonction des récompenses perçues

Fin

 Sélectionner la machine j maximisant $indice_j(t)$

 Recevoir une récompense de la machine j

Fin

La politique *UCB* a été montrée comme étant asymptotiquement optimale pour la minimisation du regret externe dans le cadre de MAB stochastique [Auer et al., 2002]. Néanmoins, la stratégie *UCB* ne nous donne aucune indication

sur l'état vers lequel elle converge. En particulier, il est prouvé par Lai et Robbins [Lai et Robbins, 1985] qu'une telle politique entraîne un regret en $O(\log T)$, où T est la longueur de l'horizon temporel. À cette fin, la politique *UCB* estime l'espérance de chaque bras d'une manière similaire à une borne supérieure de confiance.

Auer *et al.* [Auer *et al.*, 2002] ont pu prouver une borne supérieure sur l'espérance du regret après T tours en appliquant *UCB* sur K machines.



Théorème 4.2 – Théorème UCB [Auer *et al.*, 2002]

Pour tout $K > 1$, si *UCB* est exécuté sur K machines ayant des distributions de récompense arbitraires $P_1, \dots, P_K$ à support dans $[0, 1]$, alors l'espérance du regret après T tours de jeu est au plus :

$$\left(8 \sum_{i: \mu_i < \mu^*} \frac{\ln T}{\Delta_i} \right) + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j=1}^K \Delta_j \right) \quad (4.3)$$

où $\mu_1, \dots, \mu_K$ sont les espérances de $P_1, \dots, P_K$ et $\Delta_i = \mu^* - \mu_i$ et μ^* est l'espérance de la meilleure machine sur les T premiers tours.

À noter que les politiques d'*UCB* sont adaptées pour différents types d'applications. Par exemple, la résolution de problèmes d'optimisation distribués de contraintes en utilisant *UCB* a été examinée auparavant dans Ottens *et al.* [Ottens *et al.*, 2012]. Dans Kocsis et Szepesvari [Kocsis et Szepesvári, 2006], l'algorithme *UCT* (*UCB* appliqué aux arbres), a été appliqué avec succès à un problème de recherche arborescente à grande échelle, voire Gelly *et al.* [Gelly *et al.*, 2006].

Notons que les algorithmes comme *UCB* sont utilisés pour des problèmes de bandits stochastiques. Quand la fonction de récompense n'est pas stochastique (par exemple lorsque les récompenses sont attribuées par un adversaire), ces algorithmes perdent de leur efficacité. En effet, le fait que l'algorithme *UCB* soit entièrement déterministe permet à l'adversaire de prévoir ses futurs choix et donc d'attribuer des récompenses afin de forcer l'algorithme *UCB* à prendre de mauvaises décisions. Dans le cas de récompenses non stochastiques des algorithmes tels que l'algorithme *EXP3* (EXponential EXPloration-EXPloitation) [Auer *et al.*, 2003] ou Hedge [Freund et Schapire, 1995] sont utilisés. Néanmoins, nous ne nous focaliserons pas sur ce cas.

4.4

Conclusion

Nous avons présenté la dynamique de meilleure réponse où à chaque tour un unique joueur effectue une meilleure réponse, que nous avons introduite dans le chapitre 1. La dynamique de meilleure réponse est un algorithme simple qui garantit une convergence vers un équilibre de Nash dans des jeux de potentiel [Rosenthal, 1973]. Néanmoins, une telle dynamique n'assure pas de converger rapide vers un équilibre. De plus, cette convergence n'est également pas garantie dans le cas où chaque joueur effectue une meilleure réponse de manière simultanée.

Pour des jeux où tous les joueurs ont le même gain, il a été prouvé que l'algorithme *LRI* [Sastry *et al.*, 1994] converge vers un équilibre de Nash pur néanmoins sans qu'aucun temps de convergence n'ait été calculé.

Il est possible d'utiliser des algorithmes de minimisation de regret pour converger vers un équilibre de Nash. Les algorithmes de bandit manchot [Auer *et al.*, 2002, Audibert et Bubeck, 2009] sont utiles pour trouver un équilibre de Nash dans les jeux à somme nulle [Grigoriadis et Khachiyan, 1995, Audibert et Bubeck, 2009] notamment *EXP3* [Auer *et al.*, 2002]. Il a été prouvé [Grigoriadis et Khachiyan, 1995] que si les deux joueurs utilisent tous deux l'algorithme *EXP3* pour le choix de leur stratégie, alors le système converge vers un équilibre de Nash mixte.

Nous avons présenté, dans ce chapitre, divers algorithmes d'apprentissage et de minimisation de regret. Dans les chapitres suivants nous nous intéressons au calcul distribué d'équilibres dans un problème de théorie des graphes : Max-Cut, puis à une extension de ce problème pour une application à la gestion d'interférence dans des cellules mobiles. Cependant, ces différents jeux de potentiel ne nous ont pas permis de comparer les approches d'apprentissage et de minimisation de regret. Pour cela, nous nous focalisons, dans le chapitre 6, sur un autre système en concurrence : la prédiction de violation de qualité de service dans un problème de sélection d'offres de qualité de service parmi divers fournisseurs d'accès, où nous comparerons algorithmes d'apprentissage et minimisation de regret.

Calcul d'équilibres : jeu Max-Cut et généralisation à la coordination d'interférences inter cellules

Dans la première partie de ce chapitre, nous nous intéressons à un jeu issu d'un problème de théorie des graphes : Max-Cut. Le problème Max-Cut consiste à diviser en deux parties l'ensemble des sommets d'un graphe donné de façon à maximiser la somme des poids des arêtes traversant la partition. Pour ce faire, nous interprétons ces coupes comme les équilibres de Nash purs d'un jeu de potentiel.

Dans la deuxième partie de ce chapitre, nous nous intéressons à une formulation des problèmes d'interférences entre des cellules voisines dans un système de réseau mobile. Cette formulation correspond à une extension du jeu Max-Cut.

Chacune de ces parties se concentrera sur l'analyse d'un algorithme distribué pour atteindre les équilibres de Nash. Chaque joueur pourra choisir une stratégie en fonction de son histoire et de ses connaissances du système dans un jeu répété non coopératif et simultané à information incomplète.

Les travaux présentés dans ce chapitre ont donné lieu à des publications. La première publication "Distributed Selfish Algorithms for the Max-Cut Game" [Auger et al., 2013] concerne le problème de Max-Cut. Nous avons également deux publications sur la gestion de l'interférence inter cellules, en collaboration avec Amine Adouane, Kinda Kawam et Samir Tohmé, que sont "Distributed Load Balancing Game for Inter-Cell Interference Coordination" [Adouane et al., 2014a] et "Game Theoretic Framework for Inter-Cell Interference Coordination" [Adouane et al., 2014b].

Sommaire

6.1	Modèle de détection des échecs	124
6.1.1	Scénario de la négociation	126
6.1.2	Les échecs de SLA	129
6.2	Algorithme de sélection de l'offre du client	130
6.2.1	Adaptation du modèle de détection d'échec de SLA au MAB	131
6.2.2	Adaptation de la dynamique de réplication	134
6.3	Algorithme de sélection d'offres de QoS des NSPs	135
6.4	Résultats de simulations	136
6.4.1	Modèle des simulations	136
6.4.2	Résultats	138
6.5	Conclusion	143

5.1

Coupe maximale dans un graphe non pondéré

Le problème *maximum-cut* (Max-Cut) est un problème de base de la théorie des graphes, démontré comme étant NP-Complet [Karp, 1972]. Le problème Max-Cut a été largement étudié [Goemans et Williamson, 1995], même dans le cadre du calcul d'un optimum local. Parmi les différents travaux sur Max-Cut nous pouvons citer l' α_{GW} -approximation de Goemans et Williamson, avec $\alpha_{GW} = 0.878567 \dots$, qui est la meilleure approximation de Max-Cut, celui-ci ayant été prouvé inapproximable pour tout $\alpha > \alpha_{GW}$ en 1994 par Goemans et Williamson dans un premier temps [Goemans et Williamson, 1995] puis par Khot, Kindler, Mossel et O'Donnell [Khot *et al.*, 2004] en supposant que la Conjecture des Jeux Uniques (UGC) soit vraie. Il est aussi connu que la recherche d'un optimum local pour Max-Cut est PLS-complet [Schaffer, 1991] lorsque le graphe est pondéré.

Voici la définition formelle du problème Max-Cut lorsque le graphe n'est pas pondéré. Dans la suite, sauf si nous explicitons le contraire, le graphe sera considéré non pondéré.



Définition 5.1 – Max-Cut

Soit $G = (V, E)$ un graphe non orienté. Une *coupe*, notée $(S, V \setminus S)$, de G est un partitionnement de V en deux sous-ensembles non vides S et $V \setminus S$, noté $\bar{S}$ (que nous appellerons *côtés*).

La *taille* d'une telle coupe est le nombre d'arêtes ayant leurs extrémités de différents côtés soit une extrémité dans S et l'autre dans $\bar{S}$.

Le problème Max-Cut consiste à calculer une coupe dont la taille est maximisée.

Dans ce qui suit, nous cherchons à maximiser cette coupe localement de manière distribuée. Pour ce faire, nous définissons la notion de *coupe localement maximale*.



Définition 5.2 – Coupe localement maximale

Une coupe $(S, \bar{S})$ sera appelée *localement maximale* si changer de côté n'importe quel sommet unique réduit la taille de la coupe.

Soit le graphe biparti de la figure 5.1 composé de 6 sommets $\{A, B, C, D, E, F\}$. Dans les exemples suivants, nous illustrerons différents types de coupes sur ce graphe.

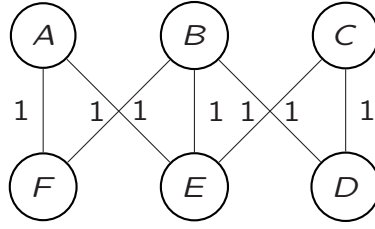


Figure 5.1 – Exemple de graphe biparti non pondéré

Nous utiliserons le code de couleurs suivant pour différencier les ensembles d'une coupe :

- En **bleu** les sommets de l'ensemble $\mathcal{S}$;
- En **cyan** les sommets de $\bar{\mathcal{S}}$.

Chaque arête appartenant à la coupe $(\mathcal{S}, \bar{\mathcal{S}})$ sera affichée en **bleu**. Les arêtes ne faisant pas partie de la coupe seront affichées en pointillés. De plus, nous noterons $w(\mathcal{S}, \bar{\mathcal{S}})$ la taille de cette coupe.

Exemple de coupes dans un graphe.

La figure 5.2 illustre la différence entre une coupe localement maximale à gauche, et la coupe optimale à droite. Dans les deux cas, étant donnée la coupe, aucun changement d'un unique sommet d'un ensemble vers l'autre ne peut augmenter sa taille.

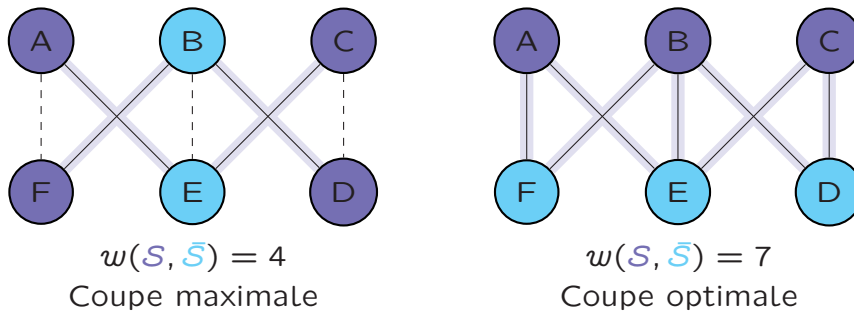


Figure 5.2 – Différence entre coupe localement maximale et coupe optimale.

Nous étudions ici le problème de Max-Cut sous forme de jeu. Ce jeu a été présenté par Christodoulou *et al.* [Christodoulou *et al.*, 2012] qui y ont étudié le temps de convergence vers un équilibre de Nash approximatif. Nous pouvons également citer les travaux de Gourvès et Monnot [Gourvès et Monnot, 2010], portant sur l'étude d'équilibres pour le jeu Max-Cut basée sur la qualité de la coupe obtenue par rapport à l'optimum, dans un contexte de communications via des signaux radio où les joueurs sont en concurrence pour deux fréquences disponibles.

Chaque sommet i du graphe est un joueur et sa stratégie est de choisir un côté, c.-à-d. $S_i = \{-1, 1\}$. Lorsque les stratégies sont fixées, nous définissons l'utilité du joueur i , notée u_i , comme étant le nombre de ses arêtes incidentes qui traversent la partition.



Définition 5.3 – Forme normale du jeu défini sur Max-Cut

Le jeu Max-Cut sous forme normale est spécifié par le triplet $G = (N, S, u)$, avec

- $N = \{1, \dots, n\}$ les sommets du graphe qui sont les joueurs ;
- $S_i = \{-1, 1\}$ l'ensemble de stratégies pures pour le joueur i ;
- u_i une fonction d'utilité pour chaque joueur i , qui à chaque profil de stratégies associe l'utilité du joueur i définie de la façon suivante :
 $u_i(s) = |\{j \in V : (i, j) \in E, s_i \neq s_j\}|$, pour un profil de stratégies $s = (s_1, \dots, s_n)$.

Dans ce jeu Max-Cut, un équilibre de Nash pur est un profil tel qu'aucun joueur ne peut changer de côté sans diminuer son utilité et donc la taille de la coupe. Il existe donc un lien entre la notion de coupe localement maximale définie précédemment (cf. définition 5.2) et l'équilibre de Nash du jeu.



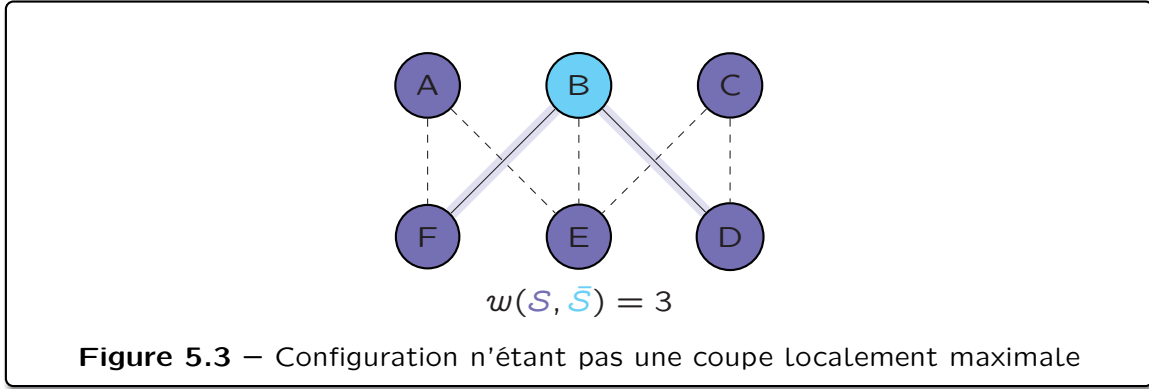
Remarque :

Une coupe localement maximale correspond à un équilibre de Nash et inversement.

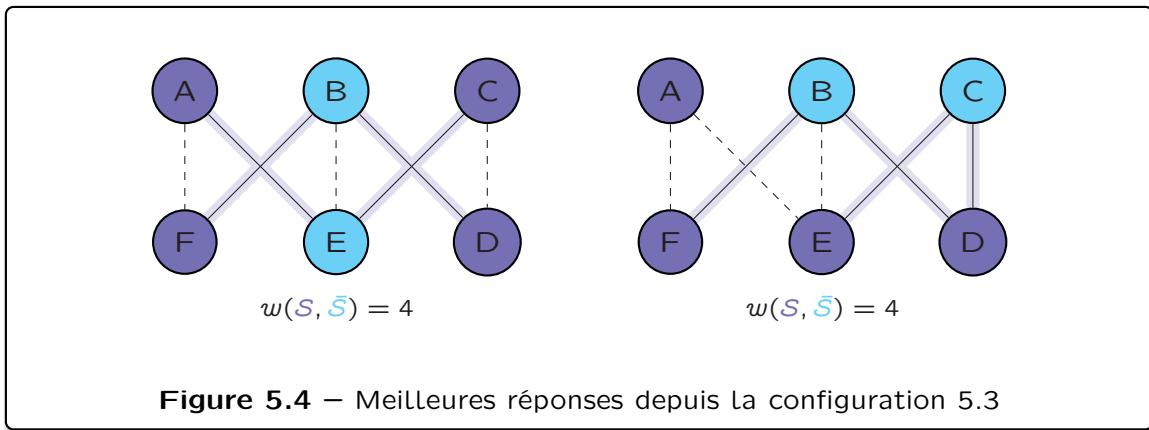
L'objectif de nos travaux est de concevoir des algorithmes distribués convergeant vers les états d'équilibre de ce jeu et l'apprentissage de ceux-ci dans ce type de situations de concurrence.

Exemple d'équilibre de Nash.

Les graphes représentés dans la figure 5.2 décrivaient une coupe maximale ainsi qu'une coupe optimale. Dans les deux cas, il n'existe aucun changement de stratégie d'un joueur permettant d'améliorer la taille de la coupe. Au contraire, d'après la configuration de la figure 5.3, il existe plusieurs mouvements permettant d'augmenter la taille de la coupe, ce qui implique que ce n'est pas un équilibre de Nash.



Comme nous pouvons le voir sur la figure 5.4 il existe deux mouvements menant à améliorer la taille de la coupe : soit en déplaçant le joueur E de l'ensemble $\mathcal{S}$ vers $\bar{\mathcal{S}}$ ou soit en déplaçant le joueur C de l'ensemble $\mathcal{S}$ vers $\bar{\mathcal{S}}$. Ce changement de stratégie correspond à une meilleure réponse pour le joueur considéré.



Le jeu Max-Cut est aussi un jeu de congestion, une sous-classe particulière de *jeux de potentiel* [Rosenthal, 1973] qui sont connus pour toujours posséder un équilibre de Nash pur et que nous avons présentés dans le chapitre 1. En effet, comme la fonction de potentiel, qui ne peut prendre qu'un nombre fini de valeurs, est strictement décroissante dans n'importe quelle séquence de mouvements de meilleures réponses pures, une telle séquence doit être finie et conduire à un équilibre de Nash [Rosenthal, 1973]. Pour le jeu Max-Cut, une fonction de potentiel simple, que nous noterons ϕ , consiste à prendre la taille de la coupe $w(\mathcal{S}, \bar{\mathcal{S}})$. C'est-à-dire, étant donné un profil s ,

$$\phi(s) = w(\mathcal{S}, \bar{\mathcal{S}}),$$

avec $\mathcal{S}$ correspondant à l'ensemble où les joueurs choisissent la stratégie 1.

Nous nous sommes intéressés au jeu Max-Cut dans sa version non pondérée. Nous avons analysé une méthode d'ordonnancement de Berenbrink, Friedetzky, Hajirasouliha et Hu [Berenbrink *et al.*, 2012], dont le modèle ainsi que les preuves peuvent être trouvés en annexe (chapitre 7). Nous avons étendu cette

analyse au cadre de notre jeu et établi un lien entre l'ordonnancement et la coupe maximale d'une clique. Cela nous a permis d'obtenir des résultats de convergence pour les graphes complets et de généraliser cette approche à des graphes quelconques. Par la suite, nous avons pu étendre cette analyse dans le cadre de la gestion d'interférence dans des réseaux mobiles.

5.1.1 Recherche d'une coupe maximale locale

5.1.1.1 Cadre de théorie des jeux

Le calcul d'un équilibre de Nash pur est connu pour être PLS-complet [Fabrikant *et al.*, 2004] dans de nombreuses classes de jeux de potentiel, y compris pour le jeu Max-Cut. En effet, calculer une coupe maximale dans un graphe pondéré est PLS-complet [Schaffer, 1991], et il existe certaines configurations à partir desquelles il faut un temps exponentiellement long pour atteindre un optimum local par de meilleures réponses successives. Chaque équilibre de Nash du jeu Max-Cut correspond à un optimum local dont le calcul est polynomial dans le cas non pondéré [Kleinberg et Tardos, 2005], mais PLS-complet en général [Schaffer, 1991].

Pour les jeux d'ordonnancement, qui sont des jeux de potentiel, le temps de convergence de la dynamique de meilleure réponse a été étudié dans les travaux [Even-Dar *et al.*, 2007]. D'autres résultats [Goldberg, 2004] de convergence dans ce modèle ont été étudiés, mais tous exigent une certaine connaissance globale du système afin de déterminer le prochain mouvement. Une version stochastique de la dynamique de meilleure réponse, où les joueurs n'ont qu'une vue locale du système, a été étudiée par Berenbrink *et al.* [Berenbrink *et al.*, 2012]. Adolphs et Berenbrink [Adolphs et Berenbrink, 2012] ont amélioré ce résultat en considérant la topologie du réseau. Christodoulou *et al.* [Christodoulou *et al.*, 2012] ont étudié le temps de convergence vers un équilibre de Nash approximatif dans le jeu Max-Cut obtenu après un nombre polynomial de meilleures réponses. Contrairement à ces travaux précédents, nous supposons que tous les joueurs sont en mesure de changer simultanément leur stratégie de manière distribuée et ont une vue locale du système.

5.1.1.2 Algorithme d'amélioration locale égoïste (LSIA)

Un algorithme simple, non distribué qui calcule une coupe localement maximale (cf. définition 5.2) consiste tout simplement à passer un par un les sommets d'un côté à l'autre tant que cela améliore la taille de la coupe, un même sommet pouvant changer plusieurs fois de côté. Lorsque ce n'est plus possible, une coupe localement maximale a été atteinte. Nous pouvons facilement en déduire qu'une coupe localement maximale peut être calculée en temps $O(|E|)$ opérations correspondant à un changement de stratégie des joueurs.

Nous notons, pour un profil de stratégies $s = (s_1, \dots, s_n)$, $r_i(s)$ la quantité $\delta_i - u_i(s)$, où δ_i est le degré du sommet i , et $u_i(s)$ le nombre d'arêtes incidentes qui traversent la coupe. Un joueur i est appelé *améliorable* s'il peut améliorer son utilité en changeant de stratégie, c.-à-d. si $r_i - u_i > 0$.

Comme décrit précédemment, nous nous concentrons sur des algorithmes distribués égoïstes où seule l'information locale est disponible. Nous présentons ici le cadre général pour un tel algorithme, appelé *Local Selfish Improvement Algorithm (LSIA)* (cf. algorithme 5.1).

Ainsi, à chaque tour, chaque joueur modifie sa stratégie indépendamment des autres joueurs. Cette modification est effectuée avec une probabilité qui ne dépend que du temps, et du nombre de ses arêtes incidentes qui traversent et ne traversent pas la partition. Notons que ces politiques sont sans mémoire, et que les joueurs sont indifférenciables.

Algorithme 5.1 – Local Selfish Improvement Algorithm (LSIA)
[Auger et al., 2013]

Données : Un graphe non orienté G non pondéré

Résultats : Une coupe localement maximale $(S, \bar{S})$

Initialisation : Choisir un profil de départ $s_0 = (s_1, s_2, \dots, s_n)$

Tant que un joueur est améliorable faire

Pour chaque joueur i en parallèle faire

 Choisir la stratégie s_i selon la Règle (cf. Règles 5.1 et 5.2) en vigueur

Fin

Fin

Nous proposons des règles pour l'algorithme LSIA (cf. Règles 5.1 et 5.2) pour des graphes complets et généraux dont l'objectif est de converger vers un équilibre en un nombre minimum d'étapes. Une *étape* correspond à un tour de jeu où l'ensemble des joueurs décident ou non de changer de stratégie.

Ce qui suit se concentre sur les graphes complets et nous prouvons que, pour une probabilité soigneusement choisie, un équilibre est atteint en $O(\log \log n)$ étapes, où n est le nombre de sommets du graphe. Par la suite, nous considérons des graphes généraux. Nous montrons que pour une probabilité fixe de changer de stratégie, l'algorithme 5.1 atteint un équilibre (c.-à-d., une coupe localement maximale) en moyenne en $4\Delta|E|$ étapes.

5.1.2 Coupe localement maximale dans un graphe complet

Même si la recherche d'une coupe maximum dans un graphe complet non pondéré n'est pas un problème en soi, lorsque nous nous plaçons dans un système distribué les sommets n'ont qu'une vision locale du système. Le choix d'une stratégie ne doit dépendre que de ce voisinage et non de l'état global

du système. Le fait de nous placer dans un graphe complet non pondéré nous permet de savoir vers quel état le système doit converger et ainsi d'évaluer l'efficacité de la méthode afin de l'étendre à des graphes quelconques.

Dans le cas des graphes complets non pondérés de n sommets, nous remarquons que la recherche d'une coupe maximale est équivalente à un ordonnancement de n tâches non pondérées sur deux ressources identiques. Ce cas particulier de jeu d'ordonnancement est analysé par la méthode proposée par Berenbrink *et al.* [Berenbrink *et al.*, 2012] et nous adaptons celle-ci.

Nous pouvons décrire comme suit pour un graphe G la Règle 5.1.

Règle 5.1 – (pour un graphe complet)

Si joueur i **est améliorable** **Alors**

| Changer de stratégie avec une probabilité $p(r_i) = 1 - \frac{n-1}{2r_i}$

Sinon

| Garder la même stratégie

Fin

Comme le graphe est complet, chaque joueur est relié aux $n - 1$ autres joueurs. L'utilité d'un joueur donné est alors égale au nombre de joueurs du graphe ayant choisi une stratégie différente de la sienne. La probabilité qu'un joueur change de stratégie est liée au nombre de joueurs se trouvant du côté différent du sien. Par conséquent, tous les joueurs d'un même côté ont en fait la même probabilité de changement de stratégie. De plus, cela implique que seuls les joueurs dans le plus grand sous-ensemble sont incités à changer de stratégie en vue d'accroître leur utilité.

Dans ce cas, nous prouvons une convergence rapide :



Théorème 5.1 – Temps de convergence vers une coupe maximale dans un graphe complet [Auger *et al.*, 2013]

Lorsque la règle 5.1 est appliquée sur un graphe complet de n sommets, une coupe localement maximale est atteinte en $O(\log \log n)$ étapes.

La suite de cette section est consacrée à la preuve de ce théorème.

Nous notons par $x(t) = \max(|S|, |\bar{S}|)$ le nombre de sommets se trouvant dans le plus grand côté au temps t . Dans le cas d'un graphe complet de n sommets, un profil de stratégies peut être simplement décrit par la valeur de $x(t)$, avec alors $x(t) \geq \frac{n}{2}$. Nous utilisons une fonction de potentiel spécifique pour le jeu Max-Cut dans un graphe complet, à savoir $\Phi(x) = (x - \frac{n}{2})^2$, que nous essayons de minimiser.

En effet, la coupe maximum d'un graphe complet $G = (V, E)$ non pondéré avec $|V| = n$ est connue. La taille de cette coupe vaut $\lfloor (\frac{n}{2})^2 \rfloor$, et est obtenue pour tout sous-ensemble $S \in V$ tel que $|S| = \lceil \frac{n}{2} \rceil$. Nous notons alors $x^* = \lceil \frac{n}{2} \rceil$ la

taille des partitions induisant une coupe maximale du graphe complet, ce qui nous donne

$$\Phi(x^*) < 1.$$



Lemme 5.1 – Convergence vers une coupe maximale

Soit $t_0 = \lceil \log \log \frac{n^2}{4} \rceil + 4$. Pour tout $t \geq 0$ et $x \in \{\frac{n}{2}, \dots, n\}$ nous avons

$$P(x(t_0 + t) \text{ est une coupe localement maximale} | x(t) = x) \geq \frac{1}{2}$$

Preuve : Soit $M(t)$ la variable aléatoire correspondant au nombre de joueurs qui changent de stratégie à l'instant t . Alors

$$\begin{aligned} \mathbb{E}[\Phi(t+1) | x(t) = x] &= \sum_{k=0}^x P(M(t) = k | x(t) = x) \Phi(x - k) \\ &= \sum_{k=0}^x P(M = k | x(t) = x) \left[\left(x - \frac{n}{2}\right)^2 - 2\left(x - \frac{n}{2}\right)k + k^2 \right] \\ &= \Phi(x) + (n - 2x)\mathbb{E}[M | x(t) = x] + \mathbb{E}[M^2 | x(t) = x] \end{aligned} \quad (5.1)$$

Conditionnée à $x(t) = x$, $M(t)$ est une variable aléatoire binomiale de paramètre p , d'où

$$\mathbb{E}[M | x(t) = x] = px$$

et

$$\mathbb{E}[M^2 | x(t) = x] = \text{Var}(M | x(t) = x) + \mathbb{E}[M | x(t) = x]^2 = xp(1-p) + (xp)^2$$

ce qui nous donne le résultat suivant, à partir de la formule (5.1) dans laquelle p est remplacé par $1 - \frac{n-1}{2(x-1)}$:

$$\begin{aligned} \mathbb{E}[\Phi(x(t+1)) | x(t) = x] &= \Phi(x) - \frac{x}{x-1} \left(\frac{n}{2} - x + \frac{1}{2} \right)^2 \\ &= \Phi(x) - \frac{x}{x-1} \left(\frac{1}{2} - \sqrt{\Phi(x)} \right)^2 \\ &\leq \Phi(x) - \left(\frac{1}{2} - \sqrt{\Phi(x)} \right)^2 \\ &\leq \sqrt{\Phi(x)} - \frac{1}{4} \end{aligned}$$

Comme cela est vrai pour tout x et par définition de l'espérance, nous en

déduisons que

$$\begin{aligned}\mathbb{E}[\Phi(x(t+1))] &\leq \mathbb{E}\left[\sqrt{\Phi(x(t))}\right] - \frac{1}{4} \\ &\leq \sqrt{\mathbb{E}[\Phi(x(t))]} - \frac{1}{4},\end{aligned}\tag{5.2}$$

La dernière inégalité résulte de l'inégalité de Jensen et du fait que la fonction racine carrée est concave.

Comme x correspond à la taille du plus grand des deux ensembles $\mathcal{S}$ et $\bar{\mathcal{S}}$, nous avons $\frac{n}{2} \leq x \leq n$ et de ce fait $\Phi(x) \leq \frac{n^2}{4}$. Cela signifie que, pour la configuration initiale, $\Phi(x(0)) \leq \frac{n^2}{4}$. Lorsque la coupe est maximale, $\Phi(x) = 0$ et dans le pire cas, $\Phi(x) = \frac{n^2}{4}$, cela permet de borner l'espérance de la fonction de potentiel.

Nous allons définir le premier τ tel qu'après τ étapes, nous avons $\mathbb{E}[\Phi(\tau)] \leq 2$. Comme $1 < \mathbb{E}[\Phi(X(\tau))] \leq \frac{n^2}{4}$ nous avons

$$\mathbb{E}[\Phi(X(\tau))] \leq \mathbb{E}[\Phi(X(0))]^{\frac{1}{2^\tau}} \Rightarrow 1 < \mathbb{E}[\Phi(X(\tau))] \leq \left(\frac{n^2}{4}\right)^{\frac{1}{2^\tau}}$$

Nous en déduisons que $\tau = \lceil \log \log \frac{n^2}{4} \rceil$ et

$$\mathbb{E}[\Phi(x(\tau)|x(0) = x)] \leq \Phi(x(0))^{2^{-\tau}} \leq 2$$

Quatre itérations supplémentaires de (5.2) nous donnent

$$\mathbb{E}[\Phi(x(t+t_0)|x(t) = x)] \leq \frac{1}{2}, \text{ avec } t_0 = 4 + \tau$$

à partir de laquelle nous obtenons le résultat de ce lemme, par l'inégalité de Markov, car une coupe localement maximale est atteinte à un moment t si $\Phi(t) < 1$. □

De plus, selon le lemme 5.1, pour chaque nouvelle exécution de t_0 étapes, la probabilité d'atteindre une coupe maximale est au moins $\frac{1}{2}$, d'où un temps de convergence vers une telle coupe d'au plus $\frac{t_0}{1-\frac{1}{2}} = 2t_0$ ce qui conclut la preuve du théorème.

5.1.3 Coupe maximum locale dans des graphes généraux

Nous utilisons la fonction de potentiel ϕ , définie dans la section 5.1, qui est la taille de la coupe étant donné un profil de stratégie $s = (s_1, \dots, s_n)$:

$$\phi(s) = |\{(i, j) \in E : s_i \neq s_j\}|.\tag{5.3}$$

Dans ce contexte, l'objectif est de maximiser cette fonction de potentiel. Ce potentiel est relié de manière simple à l'utilité des joueurs par

$$\phi(s) = \frac{1}{2} \sum_{i \in V} u_i(s)$$

Dans le cadre de l'algorithme 5.1 [Auger *et al.*, 2013], nous appliquons la règle suivante, avec Δ le degré maximum du graphe :

Règle 5.2 – (pour un graphe non pondéré)

Si le joueur i est améliorable **Alors**

| Changer sa stratégie avec une probabilité $p = \frac{1}{2\Delta}$

Sinon

| Garder la même stratégie

Fin

En appliquant cette règle à l'algorithme 5.1 dans le cas de graphes quelconques, nous pouvons prouver qu'une coupe localement maximale est atteinte. Le théorème suivant nous indique le temps moyen de convergence vers une telle coupe.



Théorème 5.2 – Temps de convergence vers une coupe maximale dans un graphe quelconque [Auger *et al.*, 2013]

Soit Δ le degré maximum du graphe $G = (V, E)$ et $p = \frac{1}{2\Delta}$ la probabilité de changement de stratégie, identique pour chaque joueur i .

Si la règle 5.2 est appliquée, une coupe localement maximale est atteinte en moyenne en $4\Delta|E|$ étapes.

Preuve : Soit $X(t)$ le profil du jeu au temps $t > 0$. Une fonction de potentiel pour ce jeu correspond au nombre d'arêtes dans la coupe au temps t . Nous notons $\phi(X(t))$ cette fonction de potentiel. Rappelons que nous notons, pour un profil de stratégies $s = (s_1, \dots, s_n)$, $r_i(s)$ la quantité $\delta_i - u_i(s)$, où δ_i est le degré du sommet i .

Soit $A(t)$ l'ensemble des sommets ayant un intérêt à se déplacer au temps t , c.-à-d. les sommets i tels que $r_i(t) - u_i(t) \geq 1$. Nous décomposons alors $A(t)$ en deux parties :

- $A_1(t)$ contient les sommets de $A(t)$ qui changent de stratégie au temps t ;
- $A_0(t) = A(t) \setminus A_1(t)$ contient les sommets de $A(t)$ qui ne changent pas de stratégie au temps t .

Les arêtes entrant ou sortant de la coupe au temps t doivent alors avoir une extrémité dans $A_1(t)$ et l'autre dans $V \setminus A_1(t)$. Notons $C(t)$ l'ensemble des arêtes dans la coupe au temps t . Nous pouvons alors déterminer $\Phi(t+1)$.

$$\begin{aligned}
\Phi(t+1) &= \Phi(t) + \sum_{i \in A_1(t)} \sum_{j \in V \setminus A_1(t)} (\mathbb{1}_{(i,j) \in C(t)} - \mathbb{1}_{(i,j) \in E \setminus C(t)}) \\
&= \Phi(t) + \sum_{i \in A_1(t)} \sum_{j \in V} (\mathbb{1}_{(i,j) \in C(t)} - \mathbb{1}_{(i,j) \in E \setminus C(t)}) \\
&\quad - \sum_{i \in A_1(t)} \sum_{j \in A_1(t)} (\mathbb{1}_{(i,j) \in C(t)} - \mathbb{1}_{(i,j) \in E \setminus C(t)}) \\
&= \Phi(t) + \sum_{i \in A_1(t)} (r_i(t) - u_i(t)) - \sum_{i \in A_1(t)} \sum_{j \in A_1(t)} (\mathbb{1}_{(i,j) \in C(t)} - \mathbb{1}_{(i,j) \in E \setminus C(t)})
\end{aligned}$$

Pour obtenir la dernière inégalité, nous utilisons le fait que $r_i(t) - u_i(t) \geq 1$ si $i \in A_1(t)$, ce qui nous donne

$$\Phi(t+1) \geq \Phi(t) + |A_1(t)| - \sum_{i \in A_1(t)} \sum_{j \in A_1(t)} \mathbb{1}_{(i,j) \in E}$$

Comme chaque sommet de $A(t)$ a une probabilité p d'être dans $A_1(t)$, indépendamment des autres sommets, et chaque sommet a un degré au plus Δ , nous en déduisons :

$$\mathbb{E}[\Phi(t+1)|X(t)] \geq \Phi(t) + p|A(t)| - p^2\Delta \cdot |A(t)|$$

En remplaçant p par $\frac{1}{2\Delta}$ nous obtenons alors $\mathbb{E}[\Phi(t+1)|X(t)] \geq \Phi(t) + \frac{|A(t)|}{4\Delta}$.

Soit $\mathcal{T}$ le temps aléatoire lorsque l'équilibre est atteint dans notre processus. Si $t \geq \mathcal{T}$, tous les sommets ne changent pas de stratégie, donc $\phi(t+1) = \phi(t)$ et $P(N > t) = 0$. Donc,

$$\begin{aligned}
\mathbb{E}[\Phi(t+1) - \Phi(t)] &= \mathbb{E}[(\Phi(t+1) - \Phi(t))\mathbb{1}_{\mathcal{T} > t}] + \mathbb{E}[(\Phi(t+1) - \Phi(t))\mathbb{1}_{t \geq \mathcal{T}}] \\
&\geq \mathbb{E}[\mathbb{E}[\phi(t+1) - \phi(t)|X(t)]\mathbb{1}_{\mathcal{T} > t}] + 0 \\
&\geq \frac{1}{4\Delta} P(\mathcal{T} > t)
\end{aligned}$$

En sommant de $t = 0$ à τ cela nous donne $\phi(\tau+1) - \phi(0) \geq \frac{1}{4\Delta} \sum_{i=1}^{\tau+1} P(\mathcal{T} > i)$.

Par conséquent, en prenant la limite des deux côtés, lorsque τ grandit

$$\phi(\mathcal{T}) - \phi(0) \geq \frac{1}{4\Delta} \sum_{i=1}^{\infty} P(\mathcal{T} > i), \text{ c.-à-d., } \phi(\mathcal{T}) - \phi(0) \geq \frac{1}{4\Delta} \mathbb{E}[\mathcal{T}]$$

d'où nous en déduisons $\mathbb{E}[\mathcal{T}] \leq 4\Delta\phi(\mathcal{T}) \leq 4\Delta |E|$ comme annoncé. $\square$

Nous n'avons pas été en mesure de trouver une borne sur le temps de convergence avec une probabilité se basant sur l'utilité des joueurs dans le cas général (qui varie au cours du temps). L'algorithme a été appliqué à différentes topologies de graphes afin de comprendre leur influence sur le temps de convergence. Dans la section suivante, nous abordons les résultats de ces simulations.

5.1.4 Autres topologies de graphes et simulations

L'algorithme 5.1 a été appliqué aux graphes généraux avec la règle 5.2. Nous avons fixé la probabilité de changer de stratégie par $p_i(t) = 1 - \frac{u_i(t)+1}{r_i(t)+1}$ pour chaque joueur i et temps t dont l'utilité est u_i et où $r_i = \delta_i - u_i$ avec δ_i le degré de i . Pour avoir une idée du temps de convergence, l'algorithme a été implémenté dans différents types de graphes : des chemins, des cycles et des graphes quelconques. Pour chaque type de graphes, les graphes considérés ont un nombre de sommets n fixé entre 10 et 5000. Chaque point de la figure est obtenu en exécutant une centaine de fois l'algorithme avec un état initial où tous les joueurs sélectionnent une stratégie uniformément au hasard.

5.1.4.1 Graphes quelconques

Nous considérons les graphes aléatoires suivants : un graphe est généré avec un nombre n fixe de sommets et une probabilité α qu'il y ait une arête entre deux sommets. Les simulations ont été faites pour plusieurs valeurs de α et les résultats obtenus sont similaires.

Dans la figure 5.5, où $\alpha = \frac{1}{2}$, nous avons pu observer avec un intervalle de confiance de 95% qu'une borne supérieure du temps de convergence pour atteindre un équilibre était moins de $n \log |E|$. Comme les graphes aléatoires utilisés sont moins denses (un petit diamètre et un grand degré), l'algorithme semble converger plus lentement que pour les graphes complets. Ce résultat souligne le fait que la topologie et la densité du graphe ont un impact sur le temps de convergence.

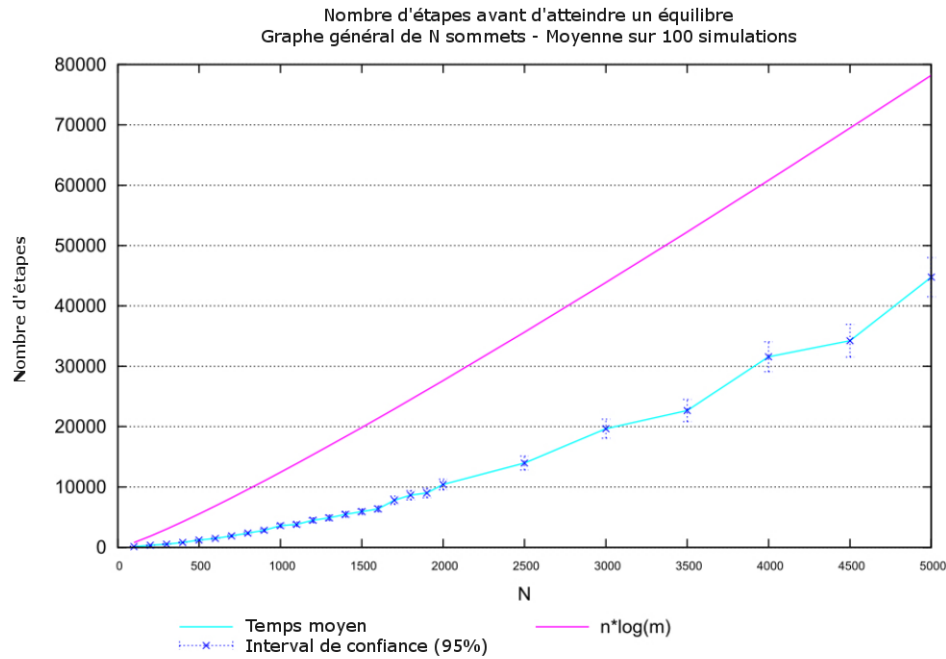


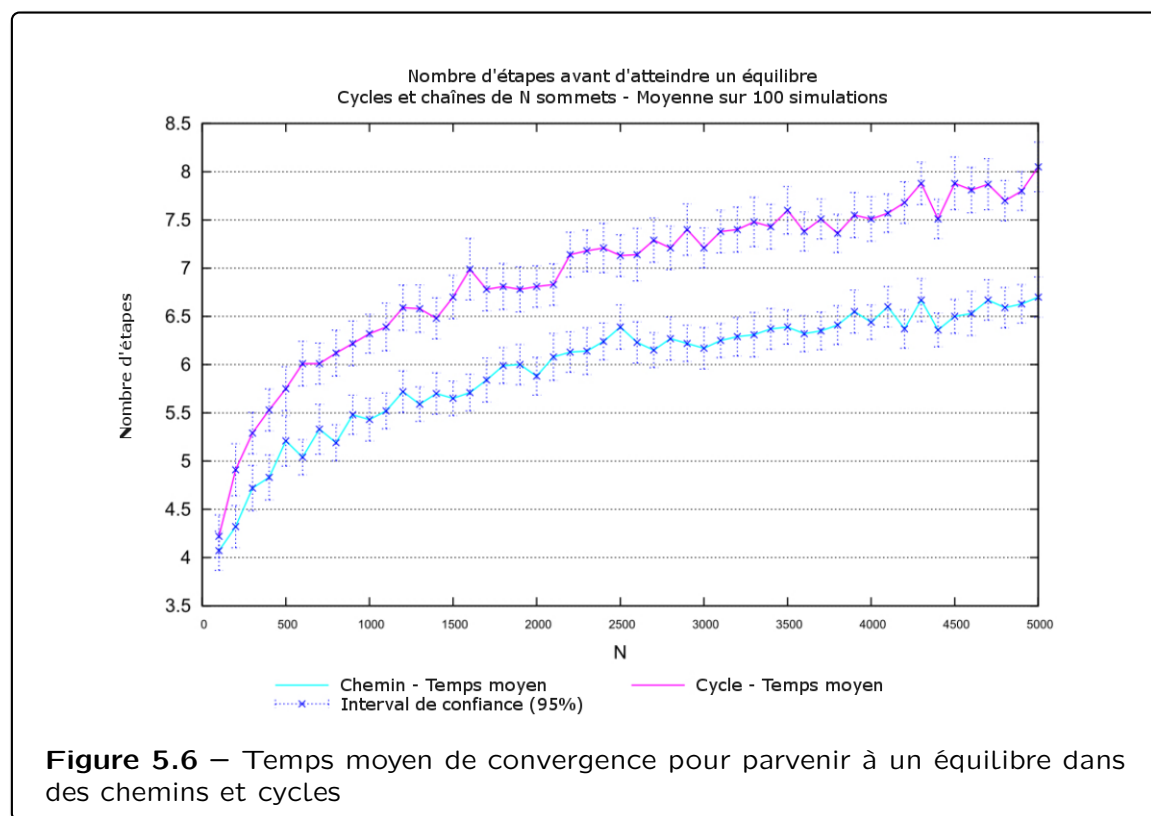
Figure 5.5 – Temps moyen de convergence pour parvenir à un équilibre dans des graphes aléatoires

5.1.4.2 Chemins et cycles

Étonnamment, pour les chemins et les cycles, le temps de convergence semble être négligeable par rapport à la taille du cycle et nous essayons de donner une idée du pourquoi. Ce résultat est lié à la topologie du graphe.

Supposons que nous nous trouvions dans un cycle. Ici, tous les joueurs ont un degré 2. Rappelons que, pour un sommet i , les stratégies possibles sont $S_i = \{1, -1\}$. Notons que pour une arête dont la configuration est $[-1, 1]$ (ou $[1, -1]$), les sommets qui lui sont incidents ne changent pas de stratégie. En effet pour les cycles, chaque joueur a la même probabilité de changement de stratégie : si $r_i(t) > u_i(t)$, alors joueur i veut changer de stratégie avec $p_i(t) = p = 1 - \frac{u_i(t)+1}{r_i(t)+1} = \frac{2}{3}$ (parce $r_i(t) = 2$ et $u_i(t) = 0$ comme le joueur i a au plus 2 voisins). L'argumentation pour le chemin est similaire, sauf que la probabilité de changer stratégie de ses deux extrémités est $\frac{1}{2}$.

Chaque fois qu'une arête avec le motif $[-1, 1]$ (ou $[1, -1]$) apparaît dans le chemin ou le cycle, chaque extrémité n'a aucun intérêt à se déplacer. Le problème est divisé en plusieurs sous-problèmes. Chaque sous-problème correspond à un chemin formé, où tous les joueurs appartiennent au même côté. Par conséquent, le temps prévu restant pour atteindre un équilibre semble être lié au temps moyen pour atteindre un équilibre dans le plus long chemin.



5.1.5 Lien avec l'interférence inter cellule

Une des applications du problème Max-Cut consiste en la gestion d'interférences entre fréquences et peut être adaptée au problème de coordination d'interférences inter cellules que nous présentons dans la section suivante. Nous pouvons en effet reprendre le jeu défini par Gourvès et Monnot [Gourvès et Monnot, 2010], modélisant la concurrence entre n joueurs communiquant via des signaux radio avec seulement deux fréquences distinctes disponibles. Le but de chaque joueur est alors de choisir sa fréquence afin de minimiser la somme des interférences qu'il éprouve.

Leurs travaux portent sur l'étude d'équilibres pour Max-Cut, mais basée sur la qualité de la coupe obtenue par rapport à l'optimum. Les équilibres obtenus, appelés *équilibres k -fort*, sont des équilibres dans lesquels aucune coalition d'au plus k joueurs ne peut, de manière unilatérale, modifier sa stratégie et augmenter de ce fait l'utilité de chacun d'eux. Cela suppose que les joueurs partagent un intérêt commun afin de former une coalition. En pratique, les joueurs sont en concurrence, ce sont des individus égoïstes dont l'intérêt est leur seul profit.

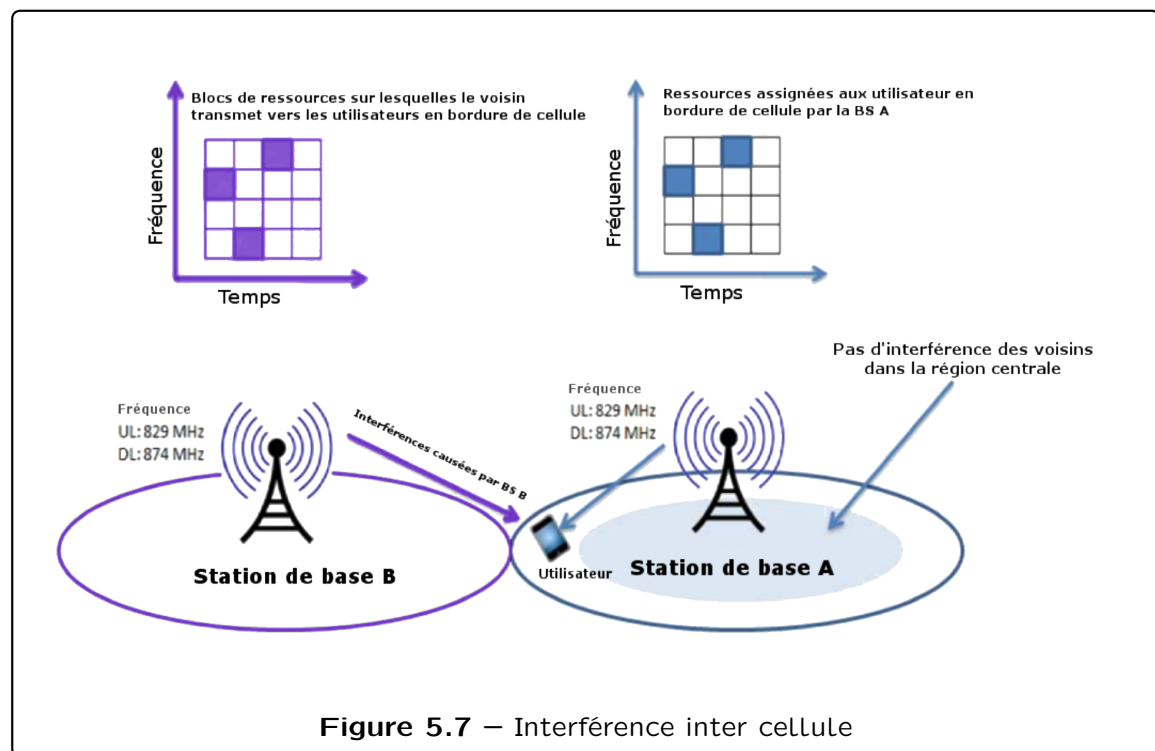
Dans le reste de ce chapitre, nous nous plaçons dans le cadre de la gestion d'interférences dans des réseaux mobiles de manière distribuée. Nous formalisons le problème d'interférences inter cellules et étendons l'analyse de l'algorithme de calcul distribué d'équilibre de Berenbrink *et al.* [Berenbrink *et al.*, 2012], sur

lequel se base l'algorithme LSIA (cf. algorithmes 5.1 et 7.2), à la résolution de celui-ci.

5.2

Coordination d'interférences inter cellules

Le problème que nous abordons consiste à minimiser les risques d'interférences entre des cellules voisines dans un système de réseaux mobiles d'un point de vue totalement distribué. Le recours à la théorie des jeux non coopératifs est tout à fait approprié pour modéliser la façon dont les *stations de base* (BSs) sont en concurrence pour des ressources limitées correspondant à des *blocs de ressources* (RBs). Nous pouvons alors modéliser la *coordination de l'interférence inter cellules* ("Inter-Cell Interference Coordination", ICIC) sous forme de jeu d'allocation de ressources.



Dans cette section, nous modélisons notre problème de gestion des ressources pour la limitation des interférences sous forme d'un jeu de potentiel, et admet donc au moins un équilibre de Nash pur, comme présenté dans le chapitre 1.

Nous adaptons une version stochastique de la dynamique de meilleure réponse à notre problème qui correspond à une généralisation du problème de Max-Cut.

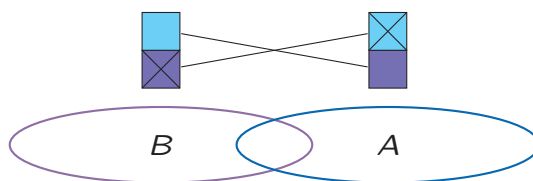


Figure 5.8 – Interprétation sous forme de graphe représentant une arête du problème de Max-Cut

La figure 5.8 illustre le problème présenté dans la figure 5.7 sous forme de Max-Cut. Nous avons ici deux stations de base A et B , reliées entre elles, et ayant chacune deux blocs de ressources (RBs) représentés par les carrés au-dessus de chaque station. Ces RBs correspondent aux deux partitions $\mathcal{S}$ et $\bar{\mathcal{S}}$ de Max-Cut, nous avons pour cela repris le code de couleur précédent, **bleu** pour $\mathcal{S}$ et **cyan** pour $\bar{\mathcal{S}}$.

Soient deux utilisateurs, un en A et un en B symbolisés par une croix. L'utilisateur en A et B a alors le choix de se placer sur l'une de ces deux partitions. Le placement de ces utilisateurs dans la figure 5.8 correspond à une coupe maximale et minimise également les interférences inter cellule (il y a interférence si deux cellules voisines utilisent les mêmes RBs). Nous pouvons étendre cette modélisation à une recherche de coupe maximale non pas sur deux partitions, mais sur k partitions, avec k le nombre de RBs de chaque station de base.

Le travail présenté ici se concentre sur des algorithmes complètement distribués. Dans un premier temps, nous décrivons le modèle du système et la caractérisation des coûts. Par la suite, nous présentons le schéma de sélection des blocs de ressources comme un jeu de potentiel non coopératif. Nous présentons l'adaptation de l'algorithme de calcul distribué de Berenbrink *et al.* [Berenbrink *et al.*, 2012], suivi de la preuve de convergence vers un équilibre de Nash. Des simulations ont également été menées pour évaluer le temps de convergence des algorithmes distribués adoptés vers des équilibres de Nash purs.

5.2.1 Modélisation du système

5.2.1.1 Contraintes réseau

Le multiplexage par répartition orthogonale de la fréquence ("Orthogonal Frequency Division Multiplexing", OFDM) est un procédé de codage de signaux numériques par répartition en fréquences orthogonales utilisé dans les réseaux mobiles de nouvelle génération (4G, "Long Term Evolution" (LTE)). Dans ces réseaux, OFDMA ("Orthogonal Frequency Division Multiple Access"), un dérivé du codage OFDM optimisé pour l'accès multiple, a été adoptée en tant que technique de couche physique clé utilisée pour l'accès radio. En effet,

elle permet une haute efficacité spectrale tout en atténuant l'interférence intracellulaire en raison de sa fonction d'orthogonalité ainsi qu'une immunité à l'atténuation sélective de fréquences et brouillage entre symboles.

Grâce à son procédé de codage par répartition en fréquences orthogonales, l'interférence intracellulaire est le plus souvent atténuée et peut être ignorée. Toutefois, lorsque la réutilisation de fréquences est déployée dans chaque cellule du système, les utilisateurs en bordure de cellule ayant une même fréquence peuvent souffrir de fortes interférences causées par les cellules voisines et une mauvaise qualité de canal, comme illustré par la figure 5.7.

Pour surmonter ce problème d'interférences, une des premières solutions proposées a été la réutilisation de fréquences fractionnaires (FFR), qui gère de façon statique la distribution de ressources de fréquences dans chaque cellule afin de diminuer l'interférence intercellulaire [Assaad, 2008]. L'interférence intercellulaire peut également être atténuée par un contrôle de puissance sur les blocs de ressources alloués avec par exemple la réutilisation des fréquences souple ("Soft Frequency Reuse", SFR) [Mao *et al.*, 2008] qui alloue une plus faible puissance pour les blocs de ressources des cellules centrales tandis que ceux des cellules en bordure se voient obtenir une puissance d'émission supérieure.

Pour réduire l'interférence intercellulaire et améliorer la performance du système, proposer des techniques efficaces de *coordination d'interférence inter cellules (ICIC)* est primordial. En général, les entités centrales effectuant les tâches de coordination d'interférence avec des connaissances globales du système doivent être évitées. Par conséquent, notre travail appartient à la catégorie des ICIC décentralisées qui est recherchée. Dans le cas de coordination d'interférences inter cellules décentralisée (par exemple, [Quek *et al.*, 2009], [Rahman *et al.*, 2009] et [Ellenbeck *et al.*, 2008]), les stations de base communiquent les unes avec les autres et optimisent leurs paramètres propres sans contrôleurs, en se basant sur de l'information locale, qui permet une adaptation rapide aux changements de situation.

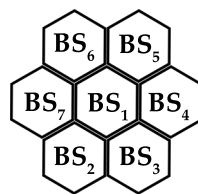


Figure 5.9 – Station de base hexagonale et son voisinage

Nous considérons un réseau cellulaire comprenant n stations de base hexagonales où $N = \{1, \dots, n\}$ représente l'ensemble des stations de base. Chacune des stations de base peut communiquer avec ses six stations de base voisines (en utilisant l'interface X2 dans LTE) comme sur la figure 5.9.

Nous nous concentrons sur le scénario de liaison descendante où OFDMA est utilisé en tant que système d'accès multiple. Les ressources de fréquences radio et temps sont regroupées en blocs de ressources temps-fréquence ("ressource blocs", RBs). Un bloc de ressource est le plus petit bloc de ressources radio qui peut être programmé pour un utilisateur mobile. Le nombre total de ces RBs est noté m . Les stations de base et les utilisateurs mobiles sont supposés avoir une unique antenne chacun. Une allocation de ressources pour une station de base i consiste à sélectionner un ensemble de RBs utilisés dans cette cellule. Pour une BS i ce choix est dénoté par le vecteur à m dimensions X_i où chaque composante $x_{i,k}$ vaut :

$$x_{i,k} = \begin{cases} 1 & \text{si le RB } k \text{ est utilisé par la BS } i, \\ 0 & \text{sinon.} \end{cases} \quad (5.4)$$

Nous utilisons le modèle de [Ellenbeck *et al.*, 2008] pour modéliser la quantité d'interférences endurées par les stations de base. Les différentes positions géographiques des utilisateurs ne sont pas prises en considération : l'utilisation de toute RB k par une BS i est affectée par l'utilisation concurrente de ce même RB par les stations de base voisines (chaque BS n'étant affectée que par ses 6 BS voisines comme dans la figure 5.9). Nous notons $N[i]$ l'ensemble des stations de base voisines de la station i . À partir de cela, nous définissons $I_{i,k}$ l'interférence endurée par une station de base i pour la RB k comme suit :

$$I_{i,k} = \sum_{\substack{j \in N[i], \\ j \neq i}} x_{j,k} \quad (5.5)$$

Nous avons présenté les contraintes techniques d'un point de vue réseau, nous allons maintenant établir la modélisation sous forme de jeu.

5.2.1.2 Jeu de concurrence entre les stations de base

La théorie des jeux non coopératifs modélise les interactions entre des joueurs en compétition pour une ressource commune. Par conséquent, il est bien adapté à la modélisation de l'ICIC. Ici, les stations de base sont les preneurs de décisions, ou joueurs, du jeu. Nous définissons un jeu de congestion multijoueurs $\mathcal{G}$ entre les n BSs. Les BSs sont supposées prendre leurs décisions sans connaître les décisions des autres. Les joueurs d'un jeu de congestion peuvent différer les uns des autres dans leurs préférences intrinsèques (le bénéfice qu'ils reçoivent de l'utilisation d'un RB spécifique), leur contribution à la congestion, ou les deux.

Nous présentons, dans ce qui suit, le cadre général du jeu. Nous notons $M = \{1, \dots, m\}$ l'ensemble des blocs de ressources disponibles et $N = \{1, \dots, n\}$ l'ensemble des joueurs que sont les stations de base. Chaque station de base i est un joueur qui doit choisir b_i blocs de ressources entre les m disponibles.

La stratégie de la station de base i est représentée par le vecteur X_i dont les composantes sont $x_{i,k}$ définies dans (5.4). Nous notons alors S_i l'ensemble des stratégies du joueur i et $S = \{S_1, \dots, S_n\}$ l'espace de tous les profils de stratégies.

Nous allons définir la fonction de coût d'une station de base i qui sélectionne la stratégie X_i comme suit :

$$\begin{aligned} c_i(X_i, X_{-i}) &= \sum_{k \in M} x_{i,k} \cdot I_{i,k} \\ &= \sum_{k \in M} \sum_{\substack{j \in N[i], \\ j \neq i}} x_{i,k} x_{j,k} \end{aligned} \quad (5.6)$$

où X_{-i} désigne le vecteur des stratégies jouées par l'ensemble des BSs, excepté la BS i .

Nous pouvons remarquer que cette fonction de coût est un cas particulier de la fonction de coût moyen du jeu de placement, définie dans le chapitre 3 par la formule 3.1. En effet $C_{moyen}(S) = \sum_{v \in V} \sum_{w \in N[v]} d_{v,w} (x_v \cdot x_w)$, en posant $d_{v,w} = 1$ et $d_{v,v} = 0$ nous obtenons bien la formule 5.6.

Chaque BS i a besoin d'une certaine quantité de RBs parmi les m RBs. Cette quantité dépend du nombre d'utilisateurs actifs dans sa cellule. En effet, sans perte de généralité, nous considérons que n'importe quel utilisateur n'est affecté qu'à un seul RB.

Notre jeu $\mathcal{G} = (N, S, c)$ peut être mis sous la forme stratégique, telle que définie dans le chapitre 1, avec

- $N = \{1, \dots, n\}$ l'ensemble des joueurs que sont les BSs ;
- S_i l'ensemble des stratégies pures pour la BS i et $S = \{S_1, \dots, S_n\}$ l'espace de tous les profils ;
- $c_i(X_i, X_{-i}) = \sum_{k \in M} \sum_{\substack{j \in N[i], \\ j \neq i}} x_{i,k} x_{j,k}$ la fonction de coût pour chaque joueur i étant donnée une stratégie X_i .

Équilibre de Nash

Notre jeu $\mathcal{G}$ est un jeu de congestion pondéré et de tels jeux n'ont en général pas de garantie d'avoir un équilibre de Nash pur. Néanmoins, ils possèdent un équilibre de Nash mixte où chaque joueur doit changer continuellement sa sélection de RB selon une probabilité de distribution sur l'ensemble des stratégies. Les équilibres mixtes conduisent à une situation où chaque utilisateur mobile doit être connecté simultanément à plus d'un RB, ce qui est interdit dans la norme LTE [LTE, 2008].

Cependant, pour notre jeu, un potentiel pondéré existe. Cela signifie que le changement unilatéral de la stratégie X_i d'un utilisateur à X'_i résulte en un

changement de sa fonction de coût qui est égal à la variation d'une fonction de potentiel $\phi : S \rightarrow \mathbb{R}$. Les jeux de potentiel admettent au moins un équilibre de Nash pur ce qui est essentiel dans le contexte actuel où un RB peut être affecté à seulement un utilisateur mobile.

5.2.2 Apprentissage distribué des équilibres de Nash purs

Nous pouvons considérer que chaque BS effectue un jeu d'ordonnancement, dont le modèle a été présenté dans le chapitre 1. En effet, dans le jeu $\mathcal{G}$ présenté précédemment, chaque BS (joueur) choisit simultanément plusieurs RBs (correspondant aux ressources dans un jeu d'ordonnancement). Par conséquent, notre adaptation des algorithmes proposés par Berenbrink *et al.* se fait facilement si nous considérons que les utilisateurs mobiles dans chaque cellule sont les joueurs considérés. Dans ce cas, notre jeu se résume à un jeu d'ordonnancement étant donné qu'un RB est alloué à un utilisateur mobile à la fois. La BS joue simplement le rôle d'une entité centrale qui interdit la sélection du même RB par différents utilisateurs mobiles. Nous proposons ci-après un algorithme adapté pour atteindre un équilibre de Nash pur où, à chaque tour, chaque BS migre au hasard et uniformément un unique utilisateur mobile d'un RB occupé vers un RB disponible. Dans ce cas, la BS n'a pas besoin de s'assurer que différents utilisateurs sont affectés au même RB. Nous reprenons l'algorithme de Berenbrink *et al.* (cf. algorithme 7.1) que nous modifions de la manière suivante :

Algorithme 5.2 – Algorithme d'allocation de RBs pour les stations de base

Initialisation : Explorer chaque ressource et recevoir la récompense

Tant que une RB est améliorable **faire**

Pour chaque BS i de 1 à n **faire**

 La BS i sélectionne un RB l au hasard de manière uniforme parmi les b_i RBs utilisés

 La BS i sélectionne un RB libre j au hasard de manière uniforme dans l'ensemble des $m - b_i$ RBs libres (stratégie x'_i)

Si $(c_i(x_i, X_{-i}) > c_i(x'_i, X_{-i}))$ **Alors**

 Changer de stratégie (aller du RB l au RB j) avec une probabilité $1 - \frac{w_j}{w_l}$, avec $w_k = \sum_{i \in N} x_{i,k}$ correspondant au nombre d'utilisateurs sur le RB k .

Fin

Fin

Fin

L'intérêt de l'utilisation d'une telle méthode réside dans le fait que les informations utilisées sont uniquement locales et déjà présentes dans le système. En effet, aucune entité centralisée n'est nécessaire. Pour calculer le coût de la stratégie antérieure (X_i) et de la stratégie actuelle (X'_i) , toute BS i se

sert d'informations de signalisation déjà présentes dans le lien descendant d'un système LTE. En effet, un utilisateur mobile attribué à un RB spécifique ℓ mesure sa qualité de canal basée sur les pilotes, c'est-à-dire de signaux cellulaires spécifiques de référence (CRS, "Cell Specific Reference Signals") qui sont répartis sur toute la bande indépendamment de l'allocation de l'utilisateur individuel. Ainsi, la fonction de coût peut être facilement déduite par l'indicateur de qualité de canal (CQI, "Channel Quality Indicator") envoyé chaque intervalle de temps de transmission (TTI, "Transmit Time Interval") par l'utilisateur mobile auquel le RB ℓ est attribué.

Dans ce qui suit, nous prouvons que cet algorithme 5.2 converge vers un équilibre de Nash pur en un temps borné.

5.2.3 Convergence vers un équilibre de Nash pur de l'algorithme

Rappelons que n est le nombre de BSs et m le nombre de RBs les ressources avec $N = \{1, 2, \dots, n\}$ et $M = \{1, 2, \dots, m\}$. De plus à chaque étape de l'algorithme 5.2, chaque BS i sélectionne un RB au hasard de manière uniforme parmi les b_i RBs utilisées, b_i désigne donc par la suite le nombre de blocs de ressources non libres pour la BS i .



Théorème 5.3 – Convergence vers un équilibre de Nash

Soit T le nombre d'étapes nécessaires pour que l'algorithme atteigne un équilibre de Nash pour la première fois, alors

$$E[T] = O(m \log(\bar{w}))$$

$$\text{avec } \bar{w} = \frac{1}{m} \sum_{i \in N} \left(\sum_{k \in M} x_{i,k} \right), \text{ c'est-à-dire } \bar{w} = \frac{n \times \sum_{i \in N} b_i}{m}.$$

Cette section est consacrée à prouver la convergence de l'algorithme 5.2.

Soit $W(t) = (w_1, w_2, \dots, w_m)$ le vecteur contenant le nombre d'utilisateurs sur chaque bloc de ressources au moment t , avec $w_k = \sum_{i \in N} x_{i,k}$ correspondant au nombre d'utilisateurs sur le RB k .

Soit $P_{k,\ell}^u$ la probabilité que l'utilisateur u se déplace du RB k vers le RB ℓ défini par :

$$P_{k,\ell}^u = \begin{cases} \frac{1}{b_u(m-(b_u-1))} \left(1 - \frac{w_\ell}{w_k}\right) & \text{if } \ell \neq k, w_k \geq w_\ell + 1 \\ 0 & \text{if } \ell \neq k, w_k \leq w_\ell + 1 \\ 1 - \sum_{j \in M, w_k > w_j} P_{k,j}^u & \text{if } \ell = k \end{cases} \quad (5.7)$$

Nous définissons notre fonction de potentiel $\Phi(W)$ comme suit

$$\Phi(W) = \sum_{k \in M} (w_k - \bar{w})^2$$

avec $\bar{w} = \frac{1}{m} \times \sum_{i \in N} \left(\sum_{k \in M} x_{i,k} \right)$ la charge moyenne.

Cette fonction de potentiel est une extension de la fonction de potentiel utilisée pour Max-Cut sur des graphes complets dans la preuve du théorème 5.1. Nous adaptons l'analyse de la convergence d'une telle fonction de potentiel proposée par Berenbrink *et al.* [Berenbrink *et al.*, 2012] à notre modèle. Pour de plus amples détails sur cette analyse, se reporter à l'annexe (cf. chapitre 7).

L'objectif de cette analyse est de montrer que pour tout temps t , l'espérance de potentiel diminue ($\mathbb{E}[\Phi(W(t+1))] < \mathbb{E}[\Phi(W(t))]$). En effet, la fonction de potentiel Φ est la somme de la différence entre la charge de chacune des ressources et la moyenne globale $\bar{w}$, le tout au carré. En supposant $W = \{\bar{w}, \bar{w}, \dots, \bar{w}\}$, ce qui signifie que toutes les ressources ont une charge égale à cette moyenne globale $\bar{w}$, alors $\Phi(W) = 0$. Ainsi, si nous prouvons $\mathbb{E}[\Phi(W(t+1))] < \mathbb{E}[\Phi(W(t))]$, étant donné que la fonction de potentiel diminue en moyenne à chaque intervalle de temps et a pour minimum $\Phi(W) = 0$, alors le temps de convergence est fini.

Afin de démontrer ce théorème, nous prouvons d'abord que $\mathbb{E}[W_k(t+1)|W(t) = w_k] < \mathbb{E}[W(t)]$. Nous définissons le potentiel moyen au temps $t+1$.

$$\begin{aligned} \mathbb{E}[\Phi(W(t+1))|W(t)] &= \mathbb{E}\left[\sum_{k=1}^m (W_k(t+1) - \bar{w})^2 | W_k(t) = w_k\right] \\ &= \sum_{k=1}^m (\mathbb{E}[(W_k(t+1)) | W_k(t) = w_k] - \bar{w})^2 \\ &\quad + \sum_{k=1}^m \text{Var}[(W_k(t+1)) | W_k(t) = w_k] \end{aligned} \quad (5.8)$$

D'après cette formule délimitant les deux sommes sur l'espérance et la variance nous arrivons à une borne sur $\mathbb{E}[\Phi(W(t+1))|W(t)]$, obtenue dans le lemme 5.4 puis raffinée dans le lemme 5.5. Le lemme 5.2 nous permet de borner le

premier terme, tandis que le Lemme 5.3 borne le second.

 **Lemme 5.2 – Borne sur** $\sum_{k=1}^m (\mathbb{E}[(W_k(t+1))|W_k(t) = w_k] - \bar{w})^2$

$$\sum_{k=1}^m (\mathbb{E}[(W_k(t+1))|W_k(t) = w_k] - \bar{w})^2 < \Phi(W) - \sum_{k=1}^m \sum_{\ell=k+1}^m 2 - \mathbb{E}[W_{k,\ell}] (w_k - w_\ell) \quad (5.9)$$

Preuve : Sans perdre de généralité, nous supposons $w_1 \geq w_2 \geq \dots \geq w_m$. $\forall k \in M$, soit $S_k(t)$ l'ensemble des utilisateurs utilisant la ressource k au temps t et u un utilisateur. Soit $\mathbb{E}[W_{k,\ell}]$ la charge moyenne totale des utilisateurs se déplaçant de la ressource k à la ressource ℓ .

$$\begin{aligned} \mathbb{E}[W_{k,\ell}] &= \sum_{u \in S_k} P_{k,\ell}^u \\ &= \left(\sum_{u \in S_k} \frac{w_k}{b_u(m - (b_u - 1))} \right) \left(1 - \frac{w_\ell}{w_k} \right) \\ &\leq \left(\frac{w_k - w_\ell}{m} \right) \end{aligned}$$

La passage à la dernière inéquation est obtenu de ce qui suit : comme $m > b_u \leq 1$, nous avons $\frac{1}{m} < \frac{1}{b_u(m - (b_u - 1))} \leq \frac{1}{m^2}$. Nous avons $\mathbb{E}[W_{k,\ell}] \leq \left(\frac{w_k - w_\ell}{m} \right)$ si $k < \ell$, $\mathbb{E}[W_{k,\ell}] = 0$ sinon. En utilisant cette définition, $\mathbb{E}[W_k(t+1)|W(t) = w_k]$ peut être redéfini comme

$$\mathbb{E}[W_k(t+1)|W(t) = w_k] = w_k + \sum_{j=1}^{k-1} \mathbb{E}[W_{j,k}] - \sum_{\ell=k+1}^m \mathbb{E}[W_{k,\ell}]$$

Ces transferts de charge peuvent être représentés par un graphe complet avec m sommets et $S = \frac{m(m-1)}{2}$ arêtes. Chacune des arêtes du graphe a un poids correspondant à la charge moyenne transférée. Soit e une arête, $e = (k, \ell)$ a un poids $\mathbb{E}[W_{k,\ell}]$ pour tout $k \in M$, $\ell \in M$. Soit $E = \{e_1, e_2, \dots, e_S\}$ avec $e_1 \leq e_2 \leq \dots \leq e_S$. Les arêtes sont activées de manière séquentielle à partir de e_1 à e_S avec e_1 celle de poids le plus faible.

Soit $W^z = \{w_1^z, w_2^z, \dots, w_S^z\}$ le vecteur de charge de chaque ressource après l'activation de z arêtes. W^0 est le vecteur W sans modification, $\forall k \in M$, $w_k^0 = w_k \Rightarrow \Phi(W^0) = \Phi(W)$. D'après la même notation W^S est le vecteur W après avoir activé S arêtes qui correspondent à tous les transferts de charge entre

les ressources possibles.

$$\begin{aligned} w_k^S &= w_k + \sum_{\ell=1}^{k-1} \mathbb{E}[W_{\ell,k}] - \sum_{j=k+1}^m \mathbb{E}[W_{k,j}] \\ &= \mathbb{E}[W_k(t+1) | W_k(t) = w_k] \end{aligned}$$

L'objectif est de mesurer la différence de potentiel entre $\Phi(W^0)$ et $\Phi(W^S)$. Soit $\Delta_z(\Phi)$ la différence de potentiel associé à l'activation de l'arête $e_z = (k, \ell)$.

$$\begin{aligned} \Delta_z(\Phi) &= \Phi(W^{z-1}) - \Phi(W^z) \\ &= \sum_{k=1}^m (w_k^{z-1} - \bar{w})^2 - \sum_{k=1}^m (w_k^z - \bar{w})^2 \\ &= 2\mathbb{E}[W_{k,\ell}] \left(w_k^{z-1} - w_\ell^{z-1} - \mathbb{E}[W_{k,\ell}] \right) \end{aligned} \quad (5.10)$$

De la formule 5.10 et de la définition précédente de W^0 et W^S , nous avons

$$\begin{aligned} \Phi(W^0) - \Phi(W^S) &= \sum_{z=1}^S \Phi(W^{z-1}) - \Phi(W^z) \\ &= \sum_{e_z \in \mathbb{E}} \Delta_z(\Phi) \end{aligned}$$

Concentrons-nous sur le moment de l'activation de l'arête $e_z = (k, \ell)$ pour déterminer $\Delta_z(\Phi)$ et donc $\Phi(W^0) - \Phi(W^S)$. D'après la formule 5.10, afin de borner la différence de potentiel $\Delta_z(\Phi)$ nous devons connaître la charge minimale de la ressource k et la charge maximale de la ressource ℓ avant l'activation de $e_z = (k, \ell)$. Pour cela nous bornons w_k^{z-1} et w_ℓ^{z-1} en estimant la charge maximale k aurait pu envoyer avant l'activation de e_z et la charge minimale que ℓ pourrait avoir reçu.

Pour toutes les ressources j avant l'activation de l'arête $e_z = (k, \ell)$ alors $\mathbb{E}[W_{k,j}] \leq \mathbb{E}[W_{k,\ell}] \leq \left(\frac{w_k - w_\ell}{m} \right)$, car $j < \ell$. La ressource k a $(m-2)$ ressources voisines différentes de ℓ . Cela implique que la charge moyenne que k pourrait envoyer à ses voisins avant d'activer $e_z = (k, \ell)$ est au plus

$$\begin{aligned} (m-2)\mathbb{E}[W_{k,\ell}] &\leq m \left(\frac{w_k - w_\ell}{m} \right) \\ w_k^{z-1} &\geq w_k - (m-2)\mathbb{E}[W_{k,\ell}] \end{aligned}$$

De la même manière, la ressource ℓ a $(m-2)$ voisins différents de la ressource k . Le poids moyen que ℓ peut recevoir d'eux avant l'activation de $e_z = (k, \ell)$ est au plus

$$w_\ell^{z-1} \leq w_\ell + (m-2)\mathbb{E}[W_{k,\ell}]$$

En utilisant ces deux résultats, nous obtenons

$$\begin{aligned}\Delta_z(\Phi) &= 2\mathbb{E}[W_{k,\ell}] \left(w_k^{z-1} - w_\ell^{z-1} - \mathbb{E}[W_{k,\ell}] \right) \\ \Delta_z(\Phi) &\geq 2\mathbb{E}[W_{k,\ell}] (w_k - w_\ell)\end{aligned}$$

À partir de ce résultat et $\Phi(W^0) - \Phi(W^S) = \sum_{e_z \in E} \Delta_z(\Phi)$ nous pouvons borner $\Phi(W^0) - \Phi(W^S)$.

$$\Phi(W^0) - \Phi(W^S) \geq \sum_{e_z \in E} 2\mathbb{E}[W_{k,\ell}] (w_k - w_\ell)$$

Nous arrivons bien au résultat de ce lemme :

$$\sum_{i=k}^m (\mathbb{E}[(W_k(t+1))|W_k(t) = w_k] - \bar{w})^2 < \Phi(W) - \sum_{k=1}^m \sum_{\ell=k+1}^m 2\mathbb{E}[W_{k,\ell}] (w_k - w_\ell) \quad (5.11)$$

□



Lemme 5.3 – Borne sur $\sum_{k=1}^m \text{Var}[(W_k(t+1))|W_k(t) = w_k]$

Soit ε un réel avec $\varepsilon > 0$, alors

$$\sum_{k=1}^m \text{Var}[(W_k(t+1))|W_k(t) = w_k] \leq (2 - \varepsilon) \sum_{k=1}^m \sum_{\ell=k+1}^m \mathbb{E}[W_{k,\ell}] (w_k - w_\ell) \quad (5.12)$$

Preuve : Rappelons que $S_k(t)$ est l'ensemble des utilisateurs sur la ressource k au temps t et u un utilisateur. Soit $Y_{(k,\ell)}^u$ le vecteur unitaire indiquant que l'utilisateur u de déplace de la ressource k à ℓ . Soit u et u' deux utilisateurs différents, alors $Y_{(k,\ell)}^u$ et $Y_{(k,\ell)}^{u'}$ sont indépendants.

$$\begin{aligned}\text{Var}[(W_k(t+1))|W_k(t) = w_k] &= \text{Var}\left[\sum_{u \in N} \sum_{u \in S_k(t)} Y_{(k,\ell)}^u\right] \\ &= \sum_{u \in N} \sum_{u \in S_k(t)} \text{Var}[Y_{(k,\ell)}^u] \\ &= \sum_{\substack{k \in M \\ k \neq \ell}} \left(\sum_{u \in S_k(t)} P_{k,\ell}^u (1 - P_{k,\ell}^u) \right) + \sum_{u \in S_\ell(t)} P_{\ell,\ell}^u (1 - P_{\ell,\ell}^u)\end{aligned}$$

Soit $\varepsilon > 0$,

$$\begin{aligned} \text{Var}[(W_k(t+1))|W_k(t) = w_k] &< \sum_{\substack{k \in M \\ k \neq \ell}} \left(\sum_{u \in S_k(t)} P_{k,\ell}^u \right) + \sum_{\ell \in S_\ell(t)} (1 - P_{\ell,\ell}^u) \\ &< \sum_{\substack{k \in M \\ k \neq \ell}} \mathbb{E}[W_{k,\ell}] \left(\frac{w_k - w_\ell}{1 + \varepsilon} \right) + \sum_{\substack{k \in M \\ k \neq \ell}} \mathbb{E}[W_{\ell,k}] \left(\frac{w_\ell - w_k}{1 + \varepsilon} \right) \end{aligned}$$

La deuxième inégalité est vraie étant donné que chaque fois qu'un utilisateur u dans le RB k a intérêt à se déplacer vers le RB ℓ , $w_\ell > w_k + 1$ cela implique $1 \leq \frac{w_k - w_\ell}{1 + \varepsilon}$, sinon $P_{\ell,k}^u = 0$.

Si $w_k > w_\ell$ alors $\mathbb{E}[W_{\ell,k}] = 0$, donc

$$\begin{aligned} \sum_{k=1}^m \text{Var}[(W_k(t+1))|W_k(t) = w_k] &= \sum_{k=1}^m \sum_{\ell \neq k} \mathbb{E}[W_{\ell,k}] \frac{w_\ell - w_k}{1 + \varepsilon} + \sum_{k=1}^m \sum_{\ell \neq k} \mathbb{E}[W_{k,\ell}] \frac{w_k - w_\ell}{1 + \varepsilon} \\ &= (2 - \varepsilon) \sum_{k=1}^m \sum_{\ell=k+1}^m \mathbb{E}[W_{k,\ell}] (w_k - w_\ell) \end{aligned} \quad (5.13)$$

□

En reprenant la formule (5.8) et le résultat des deux lemmes précédents, nous obtenons une borne sur $\mathbb{E}[\Phi(W(t+1))|W(t)]$.



Lemme 5.4 – Borne sur $\mathbb{E}[\Phi(W(t+1))|W(t)]$

$$\mathbb{E}[\Phi(W(t+1))|W(t)] < \Phi(W(t)) - \sum_{k=1}^m \sum_{\ell=k+1}^m 4\mathbb{E}[W_{k,\ell}] (w_k - w_\ell) \quad (5.14)$$

Preuve : Rappelons que $w_k > w_\ell + 1$, sinon $\mathbb{E}[W_{k,\ell}] = 0$. De ce fait, nous avons $w_k - w_\ell > 1$ et donc $w_k - w_\ell > (w_k - w_\ell - 1)$. D'après la formule 5.8 et les deux lemmes précédents, cette propriété est vraie. □

Étant donné ce résultat, nous pouvons désormais raffiner cette borne.

 **Lemme 5.5 – Borne sur $\mathbb{E}[\Phi(W(t+1))|W(t)]$**

$$\mathbb{E}[\Phi(W(t+1))|W(t)] < \Phi(W(t)) \left(1 - \frac{1}{(m+2)}\right) \quad (5.15)$$

Preuve : Par définition, $\mathbb{E}[W_{k,\ell}] = \left(\sum_{u \in S_k} \frac{w_k}{b_u(m-(b_u-1))}\right) \left(1 - \frac{w_\ell}{w_k}\right) \leq \left(\frac{w_k - w_\ell}{m}\right)$. Comme $\Phi(W) = \frac{1}{m} \sum_{k=1}^m \sum_{\ell=k+1}^m (w_k - w_\ell)^2$, nous pouvons en déduire que

$$\begin{aligned} \sum_{k=1}^m \sum_{\ell=k+1}^m \mathbb{E}[W_{k,\ell}] (w_k - w_\ell) &\geq \sum_{u \in S_k} \frac{1}{b_u(m-(b_u-1))(w_k - w_\ell)} \\ &\geq \frac{(w_k - w_\ell)^2}{m(m-1)} \end{aligned}$$

De plus, à partir du lemme 5.4, nous obtenons que

$$\mathbb{E}[\Phi(W(t+1))|W(t)] < \Phi(W(t)) - 2 \frac{m\Phi(W(t))}{b(m-b+1)}.$$

Étant donné que $m \geq b > 0$ pour n'importe $b \in M$ le nombre de RBs utilisés, nous obtenons

$$\frac{m}{b(m-b+1)} \geq \frac{2}{m}$$

ce qui mène au résultat de ce lemme. □

À partir du lemme 5.5, nous obtenons que

$$\mathbb{E}[\Phi(W(t+\tau))|W(t)] \leq \Phi(W(t)) \left(1 - \frac{1}{m}\right)^\tau.$$

De par la définition de $\Phi(W)$, cela implique $0 \leq \Phi(W) \leq \bar{w}^2$. De ce fait, après $\tau = m \log(\bar{w})$ étapes :

$$\mathbb{E}[\Phi(W(t+\tau))|W(t)] \leq \bar{w} \left(1 - \frac{1}{m}\right)^{m \log(\bar{w})} \leq 2.$$

Par l'inégalité de Markov, la probabilité que $W(t+\tau)$ soit un équilibre de Nash est supérieure à $1/2$. À partir de cela, pour chaque τ nouvelles itérations, la probabilité d'atteindre un équilibre de Nash est au moins $1/2$, ce qui implique que le temps moyen pour atteindre un tel équilibre est au plus 2τ , d'où le théorème.

Nous avons prouvé que le système convergeait vers un équilibre de Nash pur en $O(m \log(\bar{w}))$ étapes, avec $\bar{w} = \frac{1}{m} \sum_{i \in N} \left(\sum_{k \in M} x_{i,k} \right)$ pour l'algorithme 5.2.

Néanmoins, pour des systèmes disposant d'un nombre de BSs important, le temps de convergence de cet algorithme n'est pas parfait. Nous avons proposé alors un algorithme greedy (cf. algorithme 5.3), mais n'avons pas pu établir de temps de convergence pour celui-ci, c'est pourquoi nous effectuons des simulations dont les résultats se trouvent dans la section suivante.

5.2.4 Expérimentations

Les simulations ont été effectuées en collaboration avec Amine Adouane, Kinda Kawam et Samir Tohmé [Adouane *et al.*, 2014b]. Les paramètres utilisés dans ces simulations ont été choisis grâce à l'expérience de nos collaborateurs dans ce domaine.

Nous considérons un système de 19 BSs hexagonales. Nous ne considérons que les performances dans les stations de base avec 6 BSs environnantes comme sur la figure 5.9 et supposons que les BSs plus éloignées n'interfèrent pas avec elles.

Les utilisateurs sont uniformément disposés dans chaque cellule. Les BSs ont une réutilisation de fréquences de 1, avec bande passante de PRBs de 180 kHz. Les simulations ont été réalisées pour plusieurs bandes de fréquences LTE $\{3, 5, 10, 15, 20\}$ MHz. La distance entre deux BSs voisines est de 2 km et la puissance de transmission est constante et fixée à 20 Watts (43dBm). Le facteur de perte de propagation est défini sur 3.

Algorithme 5.3 – Algorithme greedy pour les stations de base

Initialisation : Explorer chaque RB et recevoir la récompense

Pour chaque *itération* **faire**

Pour i de 1 à n **faire**

Pour $l=1$ à n_i **faire**

 La BS i sélectionne un RB au hasard de manière uniforme
 parmi les $m - (l - 1)$ RBs disponibles

Fin

Si $(c_i(x_i, X_{-i}) > c_i(x'_i, X_{-i}))$ **Alors**

 Changer de stratégie avec une probabilité $1 - \frac{c_i(x'_i, X_{-i})}{c_i(x_i, X_{-i})}$

Fin

Fin

Fin

Bien que les BSs mettent à jour leurs stratégies en parallèle, nous avons remarqué, au moyen de vastes simulations, que l'algorithme 5.2, adapté de l'algorithme LSIA, a une convergence lente pour les systèmes ayant plus de 25 RBs. Par conséquent, nous proposons une autre variante greedy où, pour chaque itération, chaque BS place de façon aléatoire et uniformément ses

utilisateurs mobiles sur les RBs selon l'algorithme 5.3. Notons que n_i est le nombre d'utilisateurs mobiles présents dans la cellule de la BS i .

5.2.4.1 Vitesse de convergence

Nous avons évalué au moyen de vastes simulations la convergence de l'algorithme 5.2 et de l'algorithme 5.3 vers un équilibre de Nash. Nous sommes intéressés dans ce qui suit à l'évaluation de la vitesse de convergence des deux algorithmes. Dans la figure 5.10, le nombre moyen d'itérations pour atteindre la convergence est tracé en fonction du nombre de RBs pour une charge constante de 50%. 300 simulations ont été exécutées pour chaque scénario. Nous remarquons que, bien que l'algorithme 5.2 converge plus rapidement pour 15 RBs, l'algorithme glouton est largement plus performant pour des bandes passantes plus larges du système. La modification de plusieurs RBs à la fois permet à l'algorithme 2 de tester relativement plus de stratégies ce qui conduit à une réduction du temps de convergence.

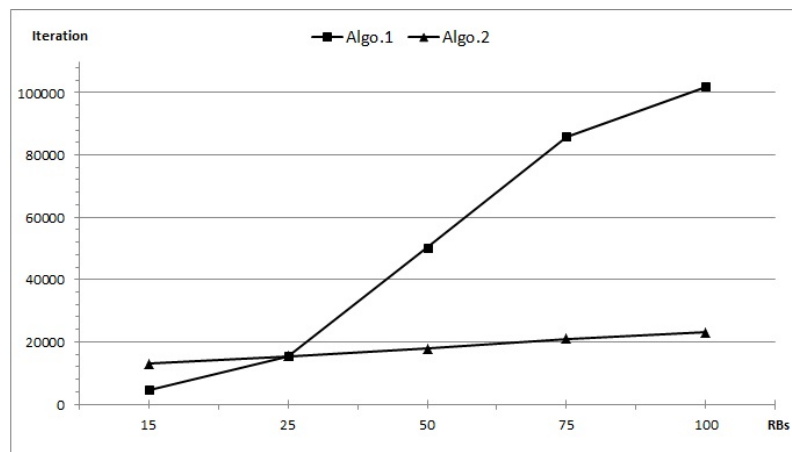


Figure 5.10 – Temps de convergence en fonction du nombre de RBs

Dans la figure 5.11, nous évaluons l'impact de la charge sur le temps de convergence. Les simulations ont été exécutées pour l'algorithme 5.3 avec une bande de fréquence de 20MHz (100RB) et une charge variant de 10% à 80%. Les résultats montrent que les temps de convergence augmentent linéairement en fonction de la charge ce qui est une propriété souhaitée.

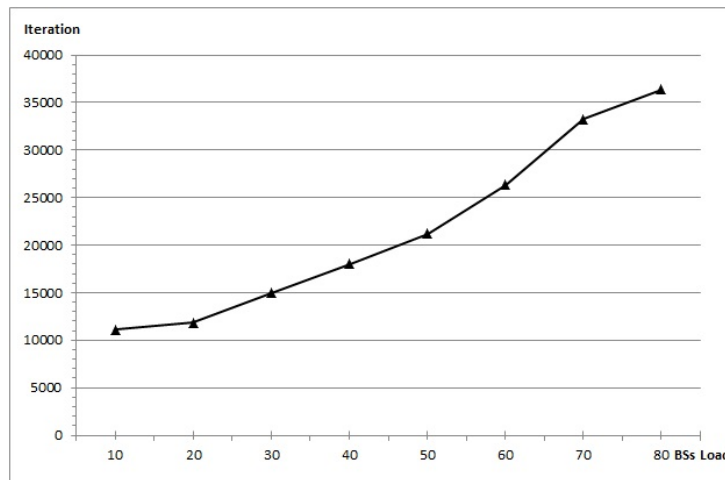


Figure 5.11 – Vitesse de convergence en fonction de la charge

5.3

Conclusion

Dans ce chapitre, nous avons étudié la convergence vers des équilibres dans un jeu basé sur le problème de Max-Cut. Le travail [Adouane *et al.*, 2014b] nous a permis d'étendre nos travaux à une application pour la gestion d'interférences inter cellules. Nous avons pu, pour les différents jeux mettre en place des algorithmes d'apprentissage afin de converger vers des états d'équilibre et prouver cette convergence.

Cependant, ces différents jeux de potentiel ne nous ont pas permis de comparer les approches d'apprentissage et de minimisation de regret. En effet lors de l'étude du problème Max-Cut nous nous sommes intéressés aux techniques de minimisation de regret présentées dans le chapitre précédent, néanmoins les résultats n'étaient pas concluants. En effet, pour Max-Cut, l'ensemble des stratégies possibles n'était que de taille 2 et il a été prouvé que lorsque le nombre d'actions est fixé par une constante le regret est borné [Nisan *et al.*, 2007]. Dans le chapitre suivant, nous étendons les travaux de Max-Cut à un problème de sélection d'offres de qualité de service parmi divers fournisseurs d'accès, où nous comparerons algorithmes d'apprentissage et minimisation de regret.

6

Minimisation de regret pour prédire la probabilité de violation de la qualité de service

Ce travail s'inscrit dans la continuité de nos travaux sur l'ordonnancement et en particulier Max-Cut. Lors de l'étude du problème Max-Cut, nous nous sommes en effet intéressés aux techniques de minimisation de regret, néanmoins les résultats étaient peu probants compte tenu du fait que pour Max-Cut l'ensemble des stratégies possibles n'était que de taille 2. Il a été prouvé que lorsque le nombre d'actions vaut 2 la borne sur le regret est de $O(\sqrt{T})$ où T est le nombre d'étapes [Nisan et al., 2007]. Pour cela, nous nous focalisons sur un autre système en concurrence : la prédiction de violation de qualité de service.

Notre système se compose d'un client ayant à faire un choix d'un fournisseur d'accès pour établir un lien vers une destination avec une certaine qualité de service. Pour cela, il dispose de plusieurs fournisseurs d'accès proposant diverses offres de qualité de service, mais, dues à des contraintes de capacités, celles-ci peuvent ne pas être disponibles à tout moment. De plus, une fois le choix du fournisseur d'accès effectué, celui-ci peut ne pas respecter la qualité de service promise qui est une violation du contrat établi.

L'objectif du client est d'apprendre de ces échecs la fiabilité des fournisseurs d'accès afin d'établir un contrat ayant une faible probabilité d'être transgressé. Pour cela, nous proposons une sélection d'offres au moyen d'algorithmes de minimisation de regret.

Les travaux présentés dans ce chapitre ont donné lieu à une publication "SLA learning from past failures, a Multi-Armed Bandit approach" [Rodier et al., 2013].

Dans le réseau Internet, de plus en plus d'applications et de services exigent un certain niveau de *qualité de service* (QoS). Nous nous concentrons sur un client nécessitant un accès à un préfixe de destination avec des garanties de qualité de service. Le client doit choisir parmi plusieurs offres de contrat proposées par des *fournisseurs d'accès réseau* ("*Network Service Providers*" ou *NSPs*). Un *accord de niveau de service* ("*Service Level Agreements*" ou *SLA*) est un contrat qui précise les garanties de qualité de service requis et les conditions tarifaires entre un fournisseur et un client. Le client n'est pas nécessairement un utilisateur final et peut être un fournisseur, un NSP, une entreprise, etc. Une fois que le client a choisi une offre, un SLA est établi entre le NSP choisi et le client.

Un accord de niveau de service ("*Service Level Agreements*" ou *SLA*) est un contrat entre un client et un *fournisseur d'accès* ("*Network Service Provider*" ou *NSP*), définissant des services pour joindre une ou plusieurs destinations (c.-à-d. préfixes IP) à une *qualité de service donnée* (QoS). Le client envoie une requête demandant une certaine qualité de service aux NSPs qui répondent avec une offre (une proposition de SLA et un prix) correspondant à ces exigences de QoS. Le client choisit parmi les offres de ces NSPs et définit un SLA avec le NSP sélectionné. Mais, une fois établi, le SLA peut ne pas être respecté, ce qui est appelé une violation, et le client n'a aucun moyen de prédire cela.

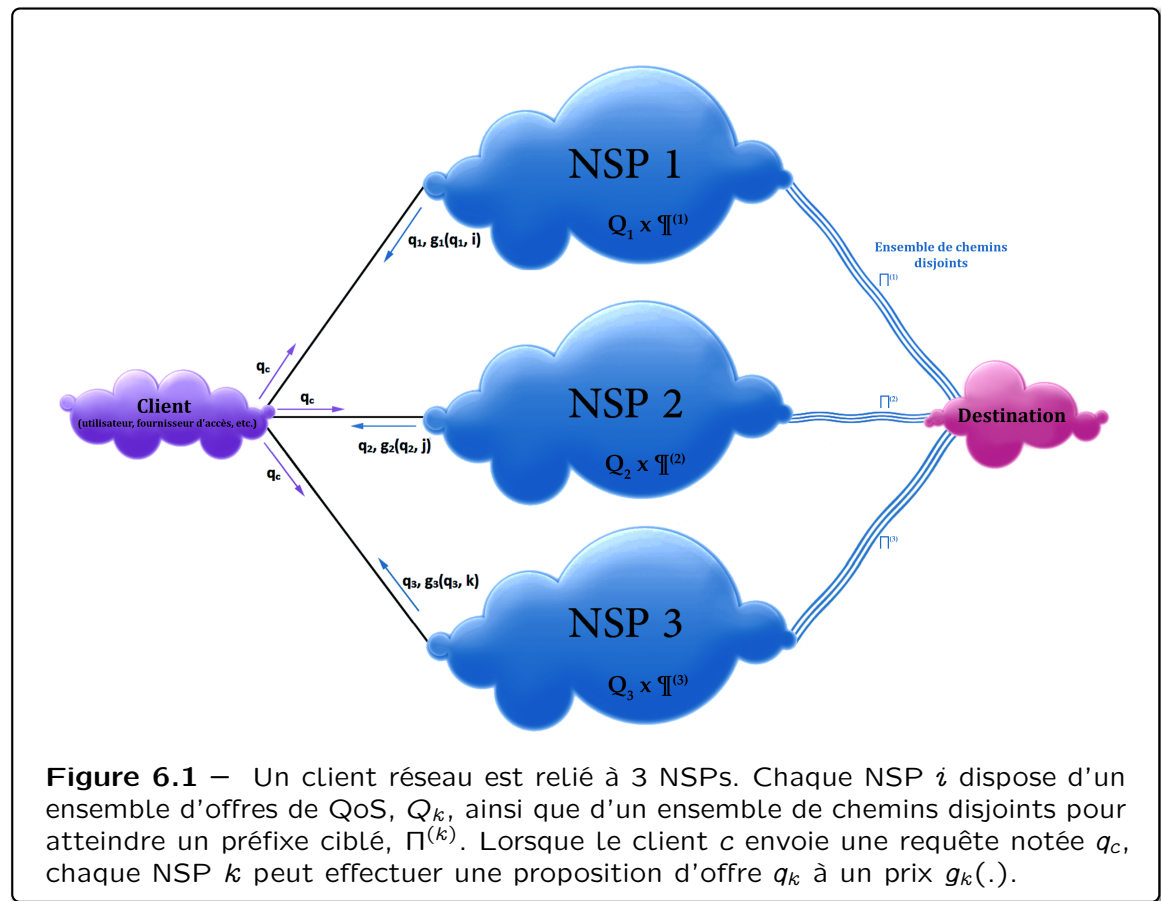
Dans un premier temps, nous détaillons le problème de négociation d'un SLA. Ce problème peut être formalisé comme un problème de Multi-Armed Bandit comme décrit dans le chapitre 4. Nous explicitons celui-ci, ainsi que les différents algorithmes utilisés par le client pour résoudre ce problème. Nous présentons des algorithmes d'apprentissage et de minimisation de regret pour le choix des offres proposées par les NSPs. Nous discutons par la suite des résultats de simulations effectués sur un exemple de réseau.

6.1

Modèle de détection des échecs

Dans le modèle décrit ci-dessous, nous nous concentrons sur un client connecté à Internet grâce à n fournisseurs d'accès réseau. Nous pouvons voir sur la figure 6.1 un exemple de système où le client est lié à 3 fournisseurs d'accès.

L'objectif du client est de sélectionner une offre, et donc un NSP, avec un bon compromis QoS/prix qui sera défini comme son gain. Le client choisit une offre parmi celles proposées par ses NSPs voisins lui accordant un bon gain et établit un SLA pour la durée demandée. Néanmoins, ce scénario n'implique pas que nous nous concentrons uniquement sur le comportement et les objectifs du client, mais sur les points suivants :



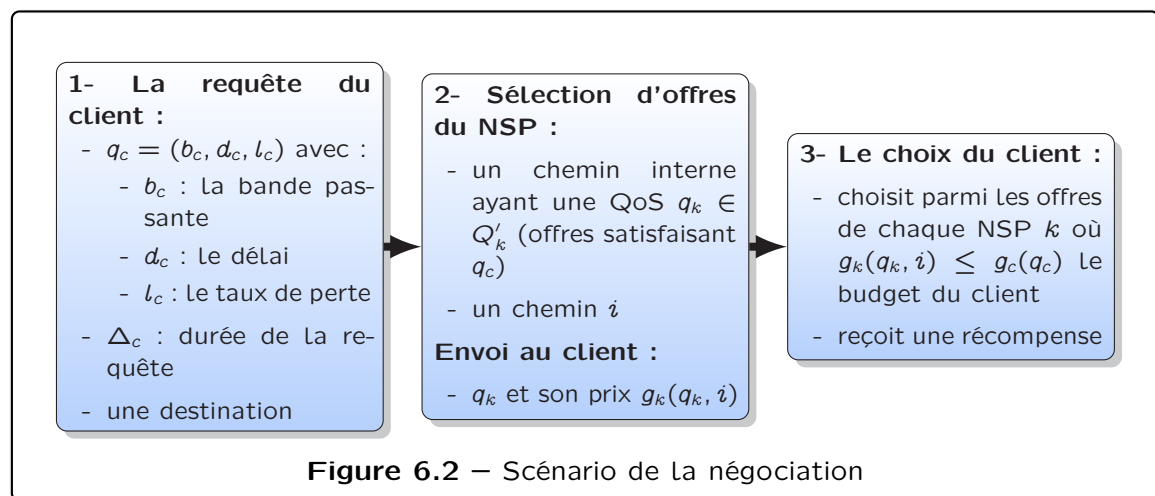
- Le respect de la qualité de service.** Nous nous concentrons sur la sélection de l'offre du client en fonction de la QoS et en tenant compte des violations de contraintes rencontrées dans le passé avec un NSP. L'objectif du client est d'évaluer correctement la fiabilité des NSPs donnés afin d'établir un SLA ayant une faible probabilité d'être transgressé. Quel que soit le client (un utilisateur final, un fournisseur, un NSP, une entreprise, etc.), cet objectif tient toujours. Par exemple, la satisfaction des utilisateurs finaux est fortement liée à la disponibilité de la destination ; l'intérêt d'un fournisseur est d'assurer la fiabilité du chemin qu'il proposera dans ses propres offres.
- Les objectifs du client.** Le client doit choisir une offre ayant une QoS satisfaisante ainsi qu'un prix compétitif. Si la qualité de service offerte et le prix sont les seuls paramètres qui affectent le gain du client, ce choix est simple et correspond à l'offre maximisant un compromis entre la QoS et le prix. Toutefois, le gain du client ne dépend pas seulement de ces deux paramètres. En effet, une fois que le client a choisi l'offre d'un NSP et établi un SLA avec lui, ce dernier pourrait ne pas être respecté et le client insatisfait. Dans ce contexte, le choix est plus complexe que l'on peut le penser. Afin de faire un bon choix, le client doit être en mesure d'estimer la fiabilité des NSPs. À cet effet, le client apprend à partir des échecs de SLAs passés en utilisant un algorithme d'apprentissage.

- **La concurrence des NSPs.** Néanmoins, les NSPs sont en concurrence les uns avec les autres ; leur objectif commun est d'être choisis par le client. La complexité du choix du NSP réside dans le fait que le NSP doit faire une offre qui est en concurrence avec celles des autres NSPs sans aucune connaissance des dites offres. Dans le cas contraire, le NSP n'est pas sélectionné et ne gagne rien. En outre, le NSP a besoin pour gérer sa capacité restante, car celle-ci affecte son taux d'échec. Le comportement du NSP généré par cette concurrence a également un impact sur le comportement du client.

6.1.1 Scénario de la négociation

Dans la section précédente, nous avons établi les objectifs des différents participants ainsi que les grandes lignes de l'établissement d'un SLA entre un client et un NSP. Le client envoie une requête demandant une certaine qualité de service aux NSPs qui répondent avec une offre (une proposition de SLA et un prix) correspondant à ces exigences de QoS. Le client choisit parmi les offres de ces NSPs et définit un SLA avec le NSP sélectionné.

La figure 6.2 résume ce scénario et les différentes notations.



Dans ce qui suit, nous détaillons les trois étapes de ce scénario.

6.1.1.1 La requête du client

La demande du client est définie par les exigences de qualité de service, une durée et un préfixe de destination. Le client fait une telle demande à chacun de ses NSPs voisins. En retour, le NSP fait sa propre offre de QoS dont le prix est établi en fonction de son état actuel (c.-à-d. de ses ressources, du marché cible, etc. tout indicateur qualifiant la valeur de la demande).

Notons que, nous n'abordons pas le problème de réservation de bande passante [Wang et Crowcroft, 1996, Douville *et al.*, 2008], nous considérons que chaque

NSP a une capacité initiale fixe qui diminue au cours de la mise en place d'un SLA. Ainsi, dans notre modélisation, l'état d'un NSP repose sur sa capacité restante, tel que défini dans les travaux de Lamali *et al.* [Lamali *et al.*, 2012].

La demande de QoS q_c du client c est un triplet $q_c = (b_c, d_c, l_c)$ dont les éléments sont les garanties requises en termes de bande passante (b_c), délai (d_c) et taux de perte (l_c). En outre, la requête q_c dispose d'une durée demandée Δ_c ainsi que d'un préfixe de destination. Le paramètre Δ_c correspond à la durée du SLA une fois établi.

Sans perte de généralité, nous supposons que le client est lié à un type unique de requête, différentes valeurs de QoS équivalant à différents clients. Ainsi, un client est *caractérisé par ses attributs de QoS*.

6.1.1.2 La réponse des NSPs

Chacun de ces NSPs a un ou plusieurs chemins disjoints vers la destination qui peuvent avoir différentes spécifications, telles que la QoS et *des taux de violation*. En outre, chaque NSP a un ensemble de chemins internes qui répondent à des vecteurs de capacités de QoS. L'offre faite au client est la combinaison de l'ensemble des chemins et toutes les capacités internes de QoS disponibles. Les NSPs sont autorisés à ne faire aucune offre, parce qu'ils ne peuvent pas atteindre le préfixe de destination par leur propre connexion, ou pas avec la QoS demandée. Nous supposons qu'ils ne motivent pas leur absence de proposition d'offre.

Chaque NSP k choisit un chemin interne et une capacité QoS de son ensemble de capacités de QoS Q_k . En outre, chaque NSP k fait une offre au client en choisissant une QoS dans un ensemble $Q'_k \subseteq Q_k$ qui satisfait les garanties de QoS demandées par le client. La section 6.3 discute des algorithmes pour déterminer de telles politiques.

Une offre est composée d'une capacité de QoS supportée $q_k \in Q'_k$, $q_k = (b_k, d_k, l_k)$ pour un NSP k et un prix $g_k(q_k, i)$ dépendant des différents paramètres de QoS de cette offre. En effet, le prix est déterminé par la formule (6.1) et reflète le coût total du chemin. La fonction permettant le calcul du prix d'une offre pour le NSP k sur le chemin i est la suivante

$$g_k(q_k, i) = p_k \times \frac{b_k}{l_k \times d_k} + g_i^{(k)}(q_k) \quad (6.1)$$

où p_k est le facteur de prix qui permet au NSP k une marge de bénéfice, et $g_i(q_k)$ la fonction de coût du chemin i pour la QoS q_k .

Ce prix est calculé à partir de la capacité de QoS q_k et le chemin i associé à cette capacité de QoS et peut-être décomposé en deux termes :

- $p_k \times \frac{b_k}{l_k \times d_k}$ correspondant au coût du chemin entre le client et le NSP ;
- $g_i^{(k)}(q_k)$ le prix du chemin entre le NSP et la destination.

Le prix offert dépend de la fonction de coût du chemin i qui est

$$g_i(q_k) = p_i \times \frac{b_k}{l_k \times d_k}$$

avec p_i étant le coût du chemin. Ceci prend en compte le prix que le NSP doit payer pour réserver le chemin i pour la QoS demandée q_k .

Si le client accepte l'offre du NSP, alors le chemin i est réservé et la capacité du NSP est réduite par b_k , la bande passante offerte. Chaque fois qu'un client envoie une requête, chaque NSP voisin a la possibilité de faire une offre selon l'algorithme choisi. Dans le cas où la bande passante de la QoS demandée par le client excède la capacité du NSP, il ne fait aucune offre. Cette supposition implique que les NPSs ne peuvent pas proposer une offre qu'ils ne sont pas capables de satisfaire.

6.1.1.3 Le choix du client

Si les NSPs peuvent ne pas faire d'offre qui ne supporte pas la QoS exigée par le client, de la même façon le client n'accepte pas d'offre dont le prix est supérieur à son budget. Le budget est calculé en utilisant l'indice de variation du prix p_c du client c et la fonction de coût g_c appliqué à la QoS demandée q_c . Une offre ayant un prix supérieur à $g_c(q_c)$, le budget du client c , sera immédiatement rejetée. Considérant la récompense du client pour une telle offre, celle-ci est fixée à 0.

Le client choisit alors parmi les offres restantes et reçoit une récompense associée. La récompense du client mesure sa satisfaction, qui doit prendre en compte le prix ainsi que les caractéristiques de la QoS offerte par rapport à celle souhaitée. Elle est définie par :

$$rew_c(k, q_k) = \begin{cases} 1 - \frac{1}{1+\beta} (\beta P_{c,k} + Q_{c,k}) \\ 0 \text{ en cas d'échec} \end{cases} \quad (6.2)$$

avec $P_{c,k} = \frac{g_k(q_k, i)}{g_c(q_c)}$ pour le prix, $Q_{c,k} = \frac{1}{3} \times (\frac{b_c}{b_k} + \frac{l_k}{l_c} + \frac{d_k}{d_c})$ pour la QoS $\beta \in 0, 1$ la sensibilité du prix dans la récompense.

Nous allons par la suite expliquer le choix de cette formule. Tout d'abord, la récompense prend en compte le ratio entre le prix de l'offre choisie et le prix que le client pourrait se permettre de payer pour la QoS qu'il a demandé, représenté par $P_{c,k}$, d'où $0 \leq P_{c,k} \leq 1$. Plus le prix de l'offre est important, plus le ratio est près du zéro. Au contraire, plus le prix est proche de la borne de prix du client, plus le ratio tend vers 1.

Le deuxième terme $Q_{c,k}$ est une normalisation de la somme de chaque composant, dans les ratios de la QoS demandée et la QoS offerte. Chaque terme $\frac{b_c}{b_k}$, $\frac{l_k}{l_c}$ et $\frac{d_k}{d_c}$ vaut 1 quand la QoS offerte est identique à la QoS demandée. Au contraire, elle est proche de 0 quand la QoS offerte est meilleure. Donc $Q_{c,k}$

est compris entre 0 et 1 et tend vers 0 quand la QoS offerte est meilleure que la QoS demandée et vers 1 dans le cas opposé.

Cette formule de récompense (formule (6.2)) ne prend pas uniquement un des aspects de l'offre en considération, prix bon marché ou meilleure QoS, mais une combinaison des deux. Le terme $P_{c,k}$ est précédé par un facteur β qui donne plus d'impact à la QoS qu'au prix dans la récompense. En effet, un petit prix indique une offre de QoS proche de la demande et vice versa, cette formule de récompense fournit un équilibre entre ces deux paramètres.

En normalisant la somme $\beta P_{c,k} + Q_{c,k}$ nous obtenons $\frac{1}{1+\beta} (\beta P_{c,k} + Q_{c,k}) = 1$ lorsque l'offre SLA est exactement la même que la demande, et tend vers 0 si meilleure.

6.1.2 Les échecs de SLA

Inspiré des travaux de Lamali *et al.* [Lamali *et al.*, 2012] notre modèle se concentre sur la détection d'erreur lors de l'établissement de SLA entre le client et un NSP. L'échec d'un SLA peut provenir :

- du NSP, avec une probabilité d'échec dépendant de sa capacité restante ;
- du chemin choisi, avec une probabilité d'échec fixe.

Pour notre étude, nous avons décidé que l'échec basé sur le chemin choisi ne dépend pas du NSP et a une probabilité fixe d'échec. Pour le taux d'échec du NSP, celui-ci est basé sur sa capacité restante, il augmente progressivement à mesure que la capacité restante du NSP diminue.

Ceci est dû aussi bien à une surcharge sur les équipements que par le fait qu'en cas d'échec d'un équipement moins de chemins alternatifs sont disponibles. Nous empruntons aux travaux de Lamali *et al.* [Lamali *et al.*, 2012] une fonction d'échec, notée $f(\cdot)$, qui dépend de la capacité actuelle NSP :

- capacité restante 0% : $f = 1$;
- capacité restante 0% à 5% : $f = 0.95$;
- capacité restante 5% à 10% : $f = 0.8$;
- capacité restante 10% à 20% : $f = 0.6$;
- capacité restante 20% à 30% : $f = 0.2$;
- capacité restante plus de 40% : $f = 0.01$.

Dans nos expérimentations, nous considérerons les deux cas :

- une violation du SLA basée sur la capacité restante du NSP avec la fonction d'échec $f(\cdot)$ et la probabilité d'échec fixe du chemin ;
- une violation du SLA basée uniquement sur la probabilité d'échec fixe du chemin choisi.

L'objectif du client est de maximiser son gain moyen et apprendre la fiabilité de ses NSPs afin d'établir un SLA ayant une faible probabilité d'être transgressé. À cette fin, nous explorons un algorithme d'apprentissage distribué d'équilibre et un algorithme de minimisation de regret que nous détaillons dans ce qui suit.

6.2

Algorithme de sélection de l'offre du client utilisant les échecs des SLAs passés

Le problème de sélection du client pourrait être considéré comme un problème de Multi-Armed Bandit (MAB) ou Bandits Manchots [Gittins et Gittins, 1979] que nous avons présenté dans le chapitre 4.

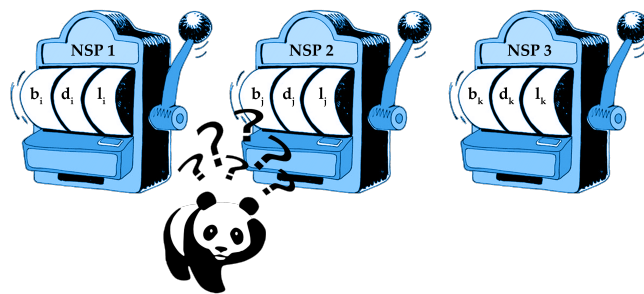


Figure 6.3 – Problème sous forme de Multi-Armed Bandit

Dans un problème classique de Multi-Armed Bandit, à chaque date de décision, un joueur choisit une action correspondant à un bras, et reçoit une récompense de celui-ci. La récompense est tirée à partir d'une distribution fixe de probabilités donnée pour le bras. Nous considérons qu'un bras correspond à un NSP. D'autres approches peuvent être envisagées, comme un bras pour chaque offre d'un NSP, néanmoins cela impliquerait de connaître l'ensemble des offres de chacun des NSPs. Par conséquent, nous établissons qu'un NSP ne propose qu'une seule offre au client à chaque fois et ce NSP correspond à un bras dans le problème de MAB.

Nous ferons l'hypothèse que la récompense (ou gain) correspondant à tous les SLA du NSP est tirée d'une *distribution de probabilité fixe*. Cette hypothèse semble être vérifiée par les résultats de simulation.

Un des algorithmes utilisés pour les problèmes de MAB que nous modifions par la suite est *l'algorithme UCB* (cf. algorithme 4.3) dont le principe est explicité dans le chapitre 4. Rappelons que si UCB est exécuté sur K machines ayant des distributions de récompense arbitraire $P_1, \dots, P_K$ à support dans $[0, 1]$, alors l'espérance du regret après T étapes de jeu est bornée (théorème 4.2).

6.2.1 Adaptation du modèle de détection d'échec de SLA au MAB

Dans un modèle de MAB du problème, le client est le joueur et il fait face aux NSPs représentant les machines. Les NSPs ont un nombre fixe d'offres avec une récompense fixée. Cependant, lors de l'établissement d'un SLA entre le client et un NSP celui-ci a une chance d'échouer. L'échec modifiant la récompense reçue par le client, chaque NSP a alors une distribution inconnue de récompense associée. En outre, lorsque le client choisit un NSP, la capacité restante de ce NSP diminue et au contraire son taux d'échec augmente. De ce fait, un NSP peut ne pas avoir une capacité restante suffisante pour proposer une offre satisfaisant le client. Ainsi, une différence notable avec les hypothèses sur lesquelles reposent les algorithmes de MAB stochastique réside dans le fait que, dans notre problème, à chaque tour, *une ou plusieurs machines peuvent ne pas être disponibles*. Pour tenir compte de cette particularité, nous avons d'abord mis l'accent sur l'algorithme UCB.

Pour appliquer un algorithme de MAB comme UCB, les NSPs doivent avoir des distributions de récompense arbitraires avec un support dans $[0, 1]$. Puisque les offres des NSPs reçues par le client sont toujours adaptées à la QoS exigée, mais peuvent avoir des prix ou garanties de QoS différentes, la récompense obtenue par le client doit prendre en compte ces deux paramètres. En ce sens, nous avons défini la fonction de la formule (6.2) pour obtenir la propriété suivante $0 \leq rew_c(k, q_k) \leq 1$.

Nous proposons une adaptation de la définition de regret dans ce contexte. Nous notons $X(t)$ l'ensemble des machines disponibles au temps t . À chaque étape t , $j \in X(t)$ si la machine j est disponible, sinon $j \notin X(t)$. Le regret après T étape sur K machines est :

$$R_T = \sum_{t=1}^T \max\{\mu_j : j \in X(t)\} - \sum_{j=1}^K \mu_j \times T_j \quad (6.3)$$

6.2.1.1 Première adaptation : UCB-Adapted

Dans un premier temps, nous avons considéré une adaptation naïve de l'algorithme *UCB*. Le problème venant du fait qu'à une étape donnée une ou plusieurs machines peuvent ne pas être disponibles, nous avons regardé le comportement de l'algorithme *UCB* si au lieu de choisir la machine offrant le

meilleur indice nous choisissons celle de meilleur indice uniquement parmi les machines disponibles.

Cette première variation de l'algorithme *UCB*, "UCB-Adapted", proposée consiste simplement à regarder à chaque étape quelles sont les machines disponibles et sélectionner la machine d'indice maximal uniquement sur cet échantillon de machines. Nous reprenons donc le calcul d'index de l'algorithme *UCB* à la différence que la sélection s'effectue sur un ensemble restreint de machines. Cela revient, pour une étape T , à sélectionner la première machine j telle que $j = \arg \max\{indice_j : 1 \leq j \leq K\}$ et j est disponible (c'est-à-dire $j \in X(T)$).

Nous appelons cet algorithme "UCB-Adapted", qui correspond à l'algorithme *UCB* (cf. algorithme 4.3) auquel nous appliquons une règle de calcul et de choix de l'indice de chaque machine (cf. Règle 6.1). Malheureusement, avec cette adaptation de l'algorithme, le théorème *UCB* sur l'espérance du regret n'est plus approprié et nous n'avons pas été en mesure de borner celui-ci.

Règle 6.1 – UCB-Adapted

Pour chaque machine j telle que $j \in X(t)$ **faire**

 Fixer $indice_j = \bar{x}_j + \sqrt{\frac{2 \ln t}{T_j}}$

Fin

Sélectionner j telle que $j \in X(T)$ et $j = \arg \max\{indice_j : 1 \leq j \leq K\}$ et recevoir la récompense

6.2.1.2 Amélioration de l'algorithme : UCB-Modified

Le problème rencontré avec l'adaptation de l'algorithme *UCB* est que nous devons prendre en compte le fait que les machines peuvent ne pas être disponibles à chacune des étapes, ce qui n'apparaît pas dans le calcul de l'indice. En effet, l'indice d'une machine calculé à une étape donnée est composé de deux éléments : la moyenne des récompenses estimée, et un facteur d'exploration. Or ce dernier dépend du nombre d'étapes jouées, ce qui peut être différent du nombre de fois où la machine a été disponible.

Afin de préserver la propriété de minimisation du regret, nous modifions le calcul de l'indice d'une machine en intégrant un nouveau facteur d'exploration (cf. formule (6.4)). Pour une étape donnée, le calcul de l'indice ne dépend plus d'un ratio $\frac{\text{nb de fois jouée}}{\text{nombre d'étapes effectuées}}$, mais du nombre de fois où le joueur a fait face à la configuration rencontrée parmi les étapes effectuées. Nous notons

$explo_j$ ce nouveau facteur d'exploration pour une machine j .

Règle 6.2 – UCB-Modified

Pour chaque machine j telle que $j \in X(t)$ **faire**

| Fixer $indice_j = \tilde{x}_j + explo_j$ (cf. formule (6.4))

Fin

Sélectionner j telle que $j = \arg \max\{indice_j : j \in X(t)\}$ et recevoir la récompense

Nous modifions l'algorithme 4.3 avec cette règle de calcul et choix d'indice suivante (cf. Règle 6.2 pour une description formelle).

Soit $X(t)$ un sous-ensemble de $\{1, \dots, K\}$ représentant l'ensemble des machines disponibles au temps t . Dans l'algorithme *UCB*, l'indice de la machine j est calculé à partir de la récompense estimée de la machine, qui ne dépend pas du temps, et un facteur d'exploration. Ce facteur d'exploration dépend du nombre de fois où la machine j a été jouée (T_j) et du nombre d'étapes où toutes les machines dans $X(t)$ ont été jouées. Nous avons alors $\sum_{i \in X(t)} T_i$ correspondant au nombre de fois où toutes les machines disponibles au temps t ont été jouées durant les T premières étapes.

Pour une machine j , soit $explo_j$ le facteur d'exploration associé à cette machine, alors :

$$explo_j = \sqrt{\frac{2 \ln \sum_{i \in X(t)} T_i}{T_j}} \quad (6.4)$$

Soit $T_X = \sum_{i \in X(t)} T_i$. En utilisant T_X à la place de T pour dénoter le nombre actuel d'étapes, l'indice calculé est le même que si le joueur faisait face à la configuration de machines $X(t)$ depuis le début jusqu'au temps T_X .

De ce fait, pour chaque sous-ensemble X de $\{1, \dots, K\}$, nous pouvons établir une borne sur l'espérance du regret en utilisant le théorème de *UCB*. De cela, on peut déduire :



Théorème 6.1 – Borne sur le regret pour UCB-Modified

Pour tout $K > 1$, si UCB-Modified est exécuté sur K machines ayant des distributions de récompense arbitraire $P_1, \dots, P_K$ à support dans $[0, 1]$, alors pour chaque sous-ensemble X de $\{1, \dots, K\}$ l'espérance du regret après T étapes de jeu est au plus :

$$\left(8 \sum_{i: \mu_i < \mu^*} \frac{\ln T_X}{\Delta_i} \right) + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j \in X}^{|X|} \Delta_j \right) \quad (6.5)$$

où $\Delta_i = \max\{\mu_j : j \in X\} - \mu_i$ pour tout $i \in X$.

6.2.2 Adaptation de la dynamique de réplication

Les modifications apportées à *UCB* ont été également appliquées à l'algorithme *LRI* afin de capturer la disponibilité des NSPs. Nous avons trois variantes de la dynamique de réplication : *LRI*, *LRI-Adapted* et *LRI-Modified*.

Le premier algorithme considéré, qui nous servira de comparaison, est l'algorithme *LRI*. L'algorithme *LRI* (cf. algorithme 4.2), présenté dans le chapitre 4, utilise un vecteur de probabilité pour la sélection de l'action suivante. Chaque action est associée à une probabilité de choix, initialement identique pour toutes les actions. Selon la récompense obtenue à chaque tour, le vecteur de probabilité est mis à jour en fonction d'un facteur $b \in [0, 1]$, d'après la formule (4.1). Nous considérons que les actions possibles sont les NSPs. Tout comme pour l'algorithme *UCB* celui-ci pourra sélectionner une action et donc un NSP n'étant pas disponible.

Dans un premier temps, nous avons considéré une adaptation naïve de l'algorithme *LRI*, que nous nommons *LRI-Adapted*. Afin de ne pas pouvoir sélectionner les NSPs non disponibles, à chaque étape, l'algorithme *LRI-Adapted* tire une offre en fonction du vecteur de probabilité, et ce jusqu'à trouver une offre dont le NSP est disponible. La règle suivante indique le choix de stratégie pour *LRI-Adapted*.

Règle 6.3 – *LRI-Adapted*

Tant que *la machine s n'est pas disponible ($s \notin X(t)$) faire*
 | Tirer une stratégie s aléatoirement en respectant le vecteur de
 | probabilités $P_\ell(t)$
Fin

Par la suite, nous considérons une règle équivalente à la précédente. Néanmoins, au lieu de tirer l'offre d'un NSP parmi l'ensemble de tous les NSPs, le vecteur est normalisé de manière à ne contenir que les NSPs disponibles à chaque étape. Nous notons cet algorithme *LRI-Modified*. Nous formalisons la règle 6.4 de la manière suivante

Règle 6.4 – *LRI-Modified*

Données : $P'_\ell(t)$: normalisation du vecteur $P_\ell(t)$ en fonction des NSPs disponibles au temps t
Pour chaque *itération t faire*
 | Calculer $P'_\ell(t)$ en fonction des NSPs disponibles au temps t
 | Tirer une stratégie s aléatoirement selon $P'_\ell(t)$
 | Recevoir un gain $gain_\ell(t) \in [0, 1]$
 | Mettre à jour les probabilités des stratégies selon la formule (4.1)
Fin

6.3

Algorithme de sélection d'offres de QoS des NSPs

Nous sommes intéressés par le comportement du client et la maximisation de ses gains cumulés. Néanmoins, la façon dont un NSP choisit une offre, pour la proposer au client, parmi son ensemble total d'offres affecte le comportement de celui-ci. Nous avons testé le comportement du client quand les NSPs utilisent trois types d'algorithmes différents. Ces types d'algorithmes sont des heuristiques naïves, une sélection aléatoire et un algorithme d'apprentissage.

Nous avons, pour un NSP k , Q_k l'ensemble des offres de QoS et Q'_k , tel que $Q'_k \subseteq Q_k$, l'ensemble des offres de QoS qui satisfont les garanties de QoS demandées par le client. Quand le client sélectionne l'offre q_k avec le chemin i proposée par le NSP k , le client obtient la récompense :

$$rew_k = \frac{g_k(q_k, i) - g_i(q_k)}{maxGain_k}$$

où $maxGain_k$ est le prix plus élevé des capacités de QoS dans Q_k . Ceci permet d'obtenir $0 \leq rew_k \leq 1$ qui est une condition nécessaire pour les algorithmes d'apprentissage décrits ci-dessus. Si le NSP n'est pas sélectionné par le client où s'il y a un échec lors de l'établissement du SLA, alors sa récompense est $rew_k = 0$.

Nous présentons ci-dessous les différents algorithmes utilisés par le NSP pour sélectionner une offre de son catalogue à proposer au client.

Heuristique : le "Discounter". Le NSP calcule le prix de toutes les offres de QoS dans Q'_k . À chaque tour, il propose au client l'offre de prix moins cher.

Heuristique : sélection aléatoire. Le NSP sélectionne une offre au hasard de manière uniforme dans l'ensemble des offres de QoS Q_k . Cet ensemble d'offres de QoS contient toutes les offres, y compris celles qui ne satisfont pas les QoS demandées par le client.

Algorithmes d'apprentissage. L'objectif est d'apprendre une politique de sélection selon les précédents échecs ou succès notés. Nous optons pour deux types d'algorithmes d'apprentissage : "Linear Reward Inaction" (LRI) [Sastry et al., 1994] et "Upper Confidence Bound" (UCB) [Auer et al., 2002].

Pour *UCB* (cf. algorithme 4.3), l'algorithme est utilisé sans modification comme décrit dans le chapitre 4. Ici, le joueur est le NSP et les machines à sous sont les offres de l'ensemble des offres de QoS Q_k . En choisissant une machine correspondant à l'offre k avec le chemin i le NSP obtient la

récompense rew_k avec $0 \leq rew_k \leq 1$. Puisque la récompense dépend de l'offre ainsi que de l'échec possible, chaque machine a une distribution de récompense inconnue.

De la même manière, l'algorithme *LRI* est utilisé sans modification comme décrit dans le chapitre 4. Les actions sont les offres de l'ensemble des capacités QoS Q_k du NSP k . Soit $P_k(t)$ le vecteur de probabilité au temps t et donc $P_{i,k}(t)$ la probabilité qu'à l'action i d'être sélectionné au temps t .

Pour toute offre $i \in Q_k$, la mise à jour du vecteur de probabilité est la suivante :

$$P_{i,k}(t+1) = P_{i,k}(t) + \begin{cases} b.rew_k(t)(P_{i,k}(t)) & \text{si } i \text{ est sélectionné} \\ -b.rew_k(t)(P_{i,k}(t)) & \text{sinon} \end{cases} \quad (6.6)$$

L'algorithme utilisé pour chaque NSP k est décrit dans l'algorithme 4.2.

6.4

Résultats de simulations

6.4.1 Modèle des simulations

Les simulations ont été réalisées sur un exemple de réseau illustré par la figure 6.1. À chaque étape de décision, le client c envoie une requête $q_c = (250\text{Mbps}, 20\text{ms}, 0.5\%)$ accompagnée d'une durée $\Delta_c = 10$. La fonction de récompense utilisée par le client est celle décrite dans la formule (6.2) avec pour paramètre $\beta = 0.5$. Étant donnés ces paramètres, la borne de prix du client calculée pour la QoS demandée vaut $g_c(q_c) = 2439.02$. De ce fait, chaque proposition de QoS faite par les NSPs dont le prix est supérieur à 2439.02 sera immédiatement rejetée.



Remarque

Les différents paramètres utilisés pour les offres de QoS et les NSPs permettent une saturation d'un des NSP qui ne peut alors plus proposer d'offre durant ces phases saturées.

Les NSPs (NSP1, NSP2 et NSP3 dans la figure 6.1) ont une capacité fixée de 3000 Mb/s. Chaque NSP dispose de deux chemins disjoints vers le préfixe de destination. Le premier chemin a un facteur de prix de 1 (type 1) tandis que le deuxième possède un facteur de prix de 5 (type 2). Tous deux ont la même capacité que celle fixée pour chacun des NSPs, de plus ils ont le même taux d'échec que le NSP auquel ils sont liés, soit : 0.1 pour le NSP1, 0.3 pour

le NSP2 et 0.6 pour le NSP3. Comme expliqué à la fin de la section 6.1, un NSP a deux types d'échecs possibles et le taux d'échec présenté ici est celui correspondant aux chemins externes des NSPs qui ont un taux fixe. Dans ce qui suit, nous considérons les échecs sur les chemins uniquement d'une part (avec une probabilité d'échec fixe pour chaque chemin) et les échecs dus à la capacité ainsi qu'aux chemins d'autre part.

Le tableau 6.1 détaille les offres de chaque NSP. Pour chaque NSP k nous présentons dans les trois premières colonnes les paramètres de bande passante (b_k), délai (d_k) et de taux de perte (l_k) des différentes offres. Induits par les valeurs sélectionnées pour les paramètres de ces offres, nous avons, pour les deux différents chemins, dans un premier temps le prix de l'offre passant par celui-ci ($g_k(q_k)$), puis la récompense du client s'il choisissait l'offre (rew_c). Ces offres de QoS ne reflètent pas des situations réelles ou autres traces, elles ont été choisies afin d'obtenir les différents comportements possibles des NSPs.

Nous ajoutons à cela 4 offres de QoS n'apparaissant pas dans le tableau 6.1, pour les NSPs 2 et 3, avec un prix qui dépasse le maximum fixé par le client. Avec ces offres, la récompense maximum que le client peut obtenir en choisissant le NSP1 est 0.407, 0.593 pour le NSP2 et 0.697 pour le NSP3.

				Chemin type 1		Chemin type 2	
	b_k	d_k	l_k	$g_k(q_k)$	rew_c	$g_k(q_k)$	rew_c
NSP1	250	20	0.5	134.14	0.315	182.92	0.308
	500	20	0.5	268.29	0.407	365.85	0.394
	2000	20	0.5	1073.17	0.381	1463.41	0.327
	3000	20	0.5	1609.75	0.317	2195.12	0.237
NSP2	750	10	0.25	804.87	0.593	1097.56	0.553
	1000	10	0.25	1073.17	0.575	1463.41	0.522
	2000	20	0.1	1094.52	0.555	1492.53	0.501
	3000	20	0.01	1649.17	0.529	2248.87	0.447
NSP3	250	5	0.01	548.90	0.642	748.50	0.615
	500	20	0.01	274.86	0.624	374.81	0.611
	750	10	0.01	824.17	0.697	1123.87	0.656

Tableau 6.1 – Les offres des NSPs choisies

Nous comparons par la suite l'algorithme de minimisation de regret *UCB* avec l'algorithme de calcul distribué d'équilibre *LRI*. Pour comparer les résultats de nos algorithmes, nous utilisons trois variantes de l'algorithme *UCB* (*UCB*, *UCB-Adapted* et *UCB-Modified*) ainsi que de l'algorithme *LRI* (*LRI*, *LRI-Adapted* et *LRI-Modified*) tous décrits dans la section 6.2. Nous comparons ces approches avec l'algorithme de meilleure réponse ainsi qu'un algorithme basé sur l'utilité (ce dernier sélectionnant une offre au hasard parmi celles ayant la plus grande utilité). La formule (6.2) est utilisée comme utilité, excepté

que celle-ci est multipliée par un facteur dénotant la réputation du NSP comme définit par Lamali *et al.* [Lamali *et al.*, 2012]. En utilisant l'algorithme avec réputation, le client prend en compte sa propre expérience concernant les échecs passés des SLAs avec les différents NSPs afin de déterminer leur réputation. Ce facteur de réputation est calculé à chaque temps t et est défini par :

$$rep_k(t) = 1 - \frac{\#echec_k(t)}{\#choix_k(t)}$$

pour le NSP k , où $\#echec_k(t)$ est le nombre de fois que le client a sélectionné une offre du NSP k qui a par la suite été transgressée, et $\#choix_k(t)$ est le nombre total de fois où les offres du NSP k ont été sélectionnées par le client.

Tous les résultats sont obtenus pour 100 simulations de 20,000 requêtes chacune. Le tableau 6.2 résume les différents scénarios utilisés, ainsi que les algorithmes de proposition d'offre des NSPs.

	NSP 1	NSP 2	NSP 3
Scenario 1	Sélection aléatoire		
Scenario 2	Le "Discounter"		
Scenario 3	Algorithme d'apprentissage : LRI		
Scenario 4	Minimisation de regret : UCB		

Tableau 6.2 – Politique de proposition d'offre des NSPs

6.4.2 Résultats

Dans un premier temps, nous comparons les différentes variantes de l'algorithme *UCB* et de *LRI* présentées dans la section 6.2. Les résultats nous indiquant que nos algorithmes *UCB-Modified* et *LRI-Modified* donnent de meilleurs résultats, nous poursuivons les simulations avec ceux-ci. Par la suite, nous comparons deux approches différentes en effectuant une comparaison entre des algorithmes de minimisation de regret avec *UCB-Modified* et d'apprentissage d'équilibres avec *LRI-Modified*.

6.4.2.1 Comparaison des algorithmes *UCB* et *LRI*

Nous fixons un scénario afin de comparer les différentes variantes de l'algorithme *UCB* et de *LRI*. Ces simulations nous permettent de nous rendre compte que, sans modification, l'algorithme *UCB* n'est pas efficace pour notre problème. Les algorithmes *UCB-Adapted* et *UCB-Modified* nous donnent de meilleurs résultats, très proches l'un de l'autre. Il en va de même pour les variantes de *LRI*. Néanmoins, comme explicité dans la section 6.2, nous avons été

en mesure d'obtenir une borne sur le regret pour l'algorithme UCB-Modified uniquement, ce qui en fait un meilleur candidat pour nos simulations.

La figure 6.4 représente le regret moyen du client sur 100 simulations. Ces simulations correspondent au scénario 1 évoqué dans le tableau 6.2 et sont exécutées pour les deux types d'échecs de SLA. Les simulations "Échec chemin uniquement" de la figure 6.4 utilisent, pour l'échec des SLAs, uniquement la probabilité d'échec des chemins externes. Pour les simulations "Échec chemin capacité et chemin", nous utilisons aussi bien la fonction d'échec (calculée pour un NSP par rapport à sa capacité restante) que les probabilités d'échec des chemins externes.

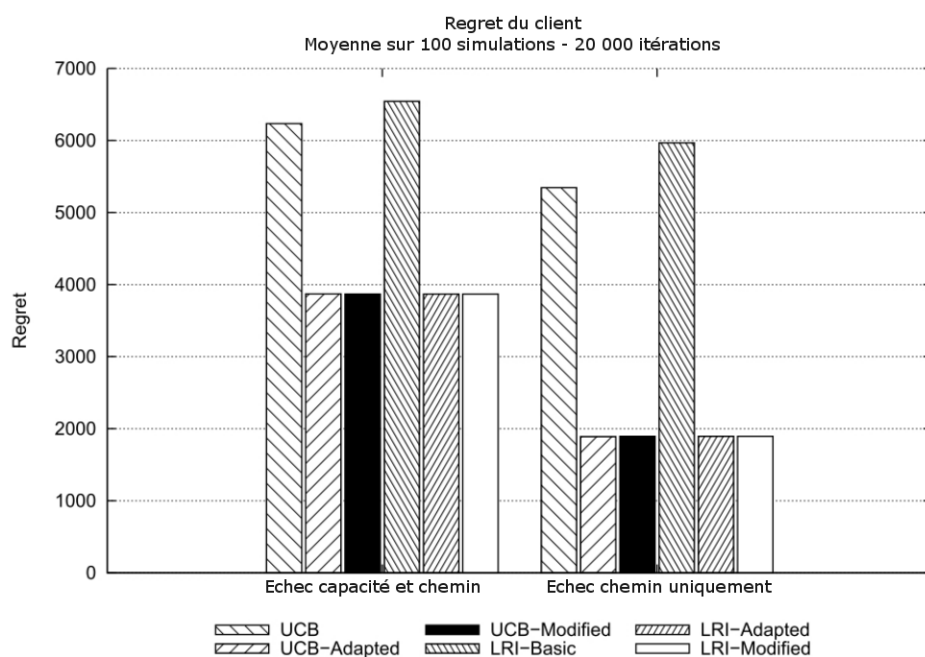


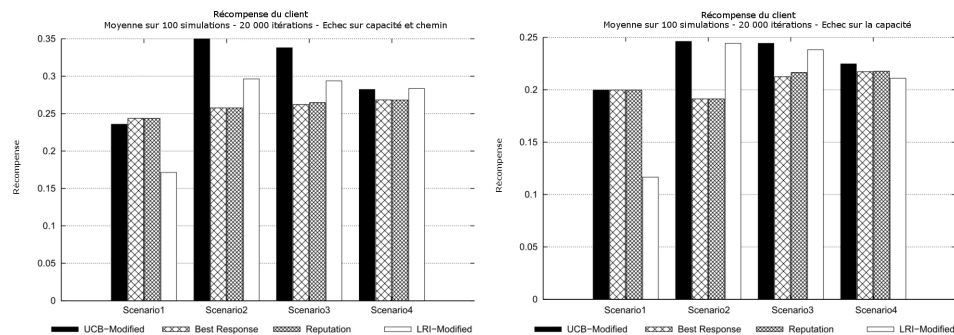
Figure 6.4 – Comparaison des algorithmes UCB dans le scénario 1 - Le regret du client

Indépendamment du cas étudié, le regret du client est moindre pour les variations de *UCB* ou *LRI* que pour l'algorithme initial. Cela est dû au fait que l'algorithme initial peut sélectionner une offre "stupide" (par exemple en sélectionnant un NSP qui n'a pas plus de capacité et ne fait pas d'offre à cette étape). Pour les deux variations de *UCB* et *LRI*, le regret ainsi que la récompense est similaire. Néanmoins, comme indiqué précédemment, la version UCB-Adapted ne permet pas de prédire ce comportement.

6.4.2.2 Comparaison entre minimisation de regret et d'apprentissage d'équilibres

Étant donnés les résultats de la figure 6.4, nous avons réalisé d'autres simulations uniquement sur les versions "Modified" des algorithmes UCB et LRI. Pour ces simulations, nous comparons l'utilisation par le client de l'algorithme

de minimisation de regret UCB-Modified avec l'algorithme d'apprentissage LRI-Modified. Afin d'avoir un point de comparaison, nous appliquons également l'algorithme de meilleure réponse (cf. chapitre 4) ainsi que l'algorithme basé sur la réputation de Lamali *et al.* [Lamali *et al.*, 2012]. Dans un premier temps, nous nous intéressons aux récompenses moyennes obtenues, puis aux taux d'échec de l'établissement de SLA.

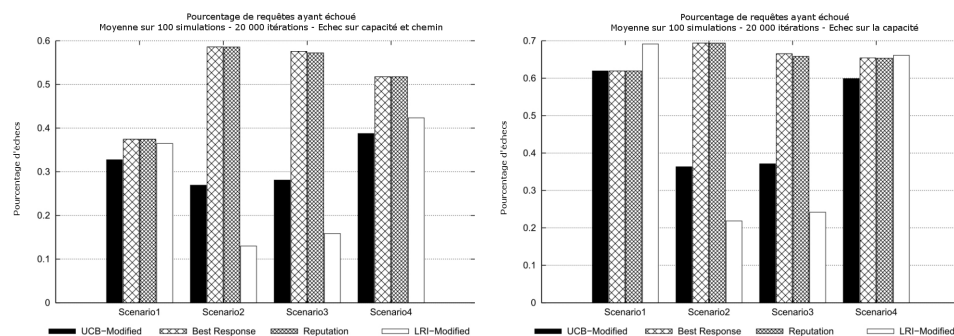


(a) Avec échec sur les chemins externes uniquement

(b) Avec échec sur les chemins externes et la capacité restante

Figure 6.5 – Récompense obtenue par le client

Nous pouvons donc examiner la capacité de l'algorithme de UCB-Modified à détecter le taux d'échec des NSPs. Pour cela, nous observons le pourcentage de demandes pour lesquelles le client a établi un SLA qui a subi un échec.



(a) Avec échec sur les chemins externes uniquement

(b) Avec échec sur les chemins externes et la capacité restante

Figure 6.6 – Pourcentage de requêtes ayant échoué

Dans les figures 6.5, les histogrammes représentent le regret cumulé moyen du

client. Quel que soit le type d'échec envisagé pour les quatre scénarios décrits dans le tableau 6.2, l'algorithme UCB-Modified donne de meilleurs résultats. Les résultats de simulation sur le scénario 2 présentent une meilleure efficacité de l'algorithme UCB-Modified. En effet, les NSPs utilisent l'algorithme Discounter et donc chaque NSP propose toujours la même offre, celle de prix le plus faible. Par conséquent, un client dont l'algorithme de sélection est basé sur l'utilité choisit toujours le même NSP ce qui augmente son taux d'échec (car réduit sa capacité restante), induisant un mauvais profit.

Les figures 6.6 montrent le pourcentage de requêtes ayant échoué. Nous remarquons que pour chaque type d'échec du SLA et de scénario du tableau 6.2, l'algorithme UCB-Modified obtient un pourcentage d'échecs plus faible que les autres.

Scénario 2								
	UCB-Mod.		Best resp.		Rep.		LRI-Mod.	
%	Sélec.	Échec	Sélec.	Échec	Sélec.	Échec	Sélec.	Échec
NSP1	58	18	0	-	0	-	88	20
NSP2	30	59	1	29	1	33	10	34
NSP3	12	67	99	69	99	69	2	60

Scénario 3								
	UCB-Mod.		Best resp.		Rep.		LRI-Mod.	
%	Sélec.	Échec	Sélec.	Échec	Sélec.	Échec	Sélec.	Échec
NSP1	56	19	1	17	1	17	96	24
NSP2	22	60	1	40	1	47	3	64
NSP3	22	63	98	67	98	66	1	60

Tableau 6.3 – Les choix du client et les échecs des NSPs - Simulations avec violation sur les chemins externes et la capacité restante

Scénario 2				
	UCB-Mod.	Best resp.	Rep.	LRI-Mod.
%	Sélect.	Sélect.	Sélect.	Sélect.
NSP1	34	0	0	88
NSP2	53	5	5	10
NSP3	12	95	95	2

Scénario 3				
	UCB-Mod.	Best resp.	Rep.	LRI-Mod.
%	Sélect.	Sélect.	Sélect.	Sélect.
NSP1	36	1	1	78
NSP2	46	7	7	12
NSP3	18	92	92	10

Tableau 6.4 – Les choix du client et les échecs des NSPs - Simulations avec violation sur les chemins externes uniquement

L'algorithme UCB-Modified est capable d'apprendre quel NSP offre la meilleure récompense avec moins de probabilité d'échouer lors de l'établissement d'un SLA. La capacité du client à apprendre la fiabilité des NSPs peut également être observé dans les tableaux 6.3 et 6.4. Le tableau 6.3 contient les résultats des simulations avec échec sur les chemins externes et la capacité restante tandis que le tableau 6.4 contient ceux avec violation sur le chemin seulement.

Ces tableaux indiquent le pourcentage de fois où chaque NSP a été sélectionné par le client et le pourcentage associé d'échecs lorsque le client utilise les algorithmes UCB-Modified, meilleure réponse, la réputation et LRI-Modified. Pour le tableau 6.4, le taux d'échec n'est pas mentionné. En effet, la violation est seulement basée sur les chemins externes et ceux-ci ont un taux d'échec fixe : 0, 1 pour le NSP1, 0, 3 pour le NSP2 et 0, 6 pour le NSP3.

Les résultats sont donnés pour les deux scénarios 2 et 3 en raison du comportement non aléatoire des NSP dans le scénario 2 et apprentissage du scénario 3. En effet, dans le scénario 2, les NSPs utilisent l'algorithme "Discounter" ce qui implique qu'ils proposent toujours la même offre, celle ayant le prix le moins cher. Pour le scénario 3, l'algorithme de sélection des NSPs est LRI-Modified qui est un apprentissage permettant de converger vers une proposition de la meilleure offre.

Rappelons que, selon le tableau 6.1, la récompense maximale obtenue par le client au moment de choisir le NSP1 est 0, 407, pour le NSP2 0, 593 et 0, 697 pour le NSP3. Quel que soit le type de violation de SLA, nous remarquons que pour les deux scénarios, lorsque le client utilise des algorithmes basés sur l'utilité, celui-ci sélectionne le NSP avec la récompense maximale, qui est le NSP3, mais c'est également celui dont le taux d'échec est le plus important. En revanche, lorsque le client utilise des algorithmes d'apprentissage tels que UCB-Modified et LRI-Modified, nous pouvons remarquer qu'un compromis entre la récompense et le taux d'échec est obtenu. En effet, nous observons dans le tableau 6.3 que le client sélectionne le NSP1 qui a le taux d'échec minimum parmi l'ensemble des NSPs.

Dans le tableau 6.4 nous pouvons remarquer un comportement différent pour les algorithmes LRI-Modified et UCB-Modified. L'algorithme LRI-Modified converge vers la sélection du NSP1 qui a le plus faible taux d'échec, tandis que l'algorithme UCB est partagé entre le NSP2 et le NSP1. Cela est dû à la manière dont l'algorithme LRI met à jour progressivement son vecteur de probabilité, pour la sélection de NSP, sur la base des récompenses précédentes. En effet lorsqu'un NSP est sélectionné la composante du vecteur de probabilité est augmentée proportionnellement à la récompense obtenue. Plus un NSP est sélectionné, plus sa probabilité augmente jusqu'à tendre vers 1. Le fait que l'une des composantes du vecteur atteint 1 "fige" l'algorithme de sélection sur l'action correspondant à celle-ci, c'est-à-dire qu'il ne peut plus changer de décision et propose toujours la même offre.

Au contraire, UCB-Modified n'est pas fixé de façon permanente sur le choix d'un

NSP, mais utilise toutes les informations sur les récompenses précédentes afin de faire son choix. Nous obtenons un compromis entre le NSP2 qui présente un taux d'échec légèrement supérieur au NSP1, mais équilibré par une récompense plus attrayante.

En conclusion, ces résultats renforcent la croyance que les algorithmes de minimisation de regret permettent au client d'évaluer la fiabilité des NSPs et sont plus adaptés à des situations dynamiques.

6.5

Conclusion

Dans ce chapitre, nous avons fourni un modèle pour le problème rencontré par un client au moment de choisir une offre de QoS qui peut rencontrer des échecs dans un contexte où la fiabilité des fournisseurs peut changer (en raison de la capacité ou d'autres contraintes, ils ne font pas des offres). Ce problème est équivalent à celui d'un joueur ayant une utilité variable face à des machines dynamiques dans un problème de Multi-Armed Bandit.

Nous avons proposé une adaptation de deux algorithmes très présents dans la littérature, *LRI* et *UCB*, pour résoudre ce problème. Les résultats obtenus renforcent la croyance que les algorithmes de minimisation de regret sont de bons candidats pour l'étude de la fiabilité des NSPs. La dynamique de réplification nous donne également de bons résultats, mais ses propriétés de calcul d'équilibre la faisant se figer sur une stratégie au bout d'un certain temps la rendent moins adaptée à des situations dynamiques.

Comme nous l'avons explicité, la tendance de l'algorithme *LRI* à se figer sur la stratégie qu'il juge optimale à un certain temps n'en fait pas un bon candidat pour des jeux en constante évolution. Au contraire, les algorithmes de minimisation de regret capturent bien cette problématique, mais ils ne garantissent pas la convergence vers un équilibre de Nash pur. Les algorithmes que nous avons proposés sont adaptés à une telle configuration (par exemple, ils détectent les comportements d'actualisation). Cette première étude nous semble prometteuse. Pour de futurs travaux, nous pouvons nous intéresser à une comparaison de divers algorithmes de minimisation de regret avec par exemple EXP3, pour lequel il a été prouvé que le système converge vers un équilibre de Nash mixte [Grigoriadis et Khachiyan, 1995].

Conclusion

Dans ce document, nous nous sommes focalisés sur les jeux de congestion, qui sont utilisés pour modéliser de nombreux systèmes en concurrence dans les réseaux de télécommunications [Ellenbeck *et al.*, 2008, Adouane *et al.*, 2014b] ou dans l'ordonnancement [Berenbrink *et al.*, 2012, Even-Dar *et al.*, 2007, Christodoulou *et al.*, 2012, Goldberg, 2004]. Nous avons généralisé ces jeux à un jeu que nous avons appelé *jeu de placement*, dont nous avons donné un exemple classique d'application en introduction avec le placement d'un individu dans un train. La première partie de ce document se concentre sur l'analyse des propriétés de ce jeu de placement, tandis que la seconde partie est consacrée au calcul distribué d'équilibres dans d'autres jeux de potentiel.

Les concepts fondamentaux de la théorie des jeux, et plus particulièrement les jeux de potentiel, qui sont une classe de jeux ayant pour particularité de toujours posséder un équilibre de Nash, ont été présentés dans le chapitre 1.

Le chapitre 2 s'est consacré à la formalisation de notre jeu de placement qui est une généralisation d'un jeu d'ordonnancement dans un graphe où le placement du joueur ainsi que son voisinage influent sur son bien-être. Nous avons introduit dans ce chapitre plusieurs cas d'étude du jeu de placement qui correspondent à des changements topologiques du graphe associé au jeu. Parallèlement, nous avons illustré ces cas d'étude par de petits exemples permettant d'exhiber différentes propriétés de ces jeux et avons vu que certaines variantes de ce jeu ne possèdent pas d'équilibre de Nash. Nous nous sommes alors posé la question fondamentale qui est de savoir si l'existence d'équilibre de Nash était garantie dans le cadre général de notre jeu ou ses différentes variantes et si tel est le cas s'ils étaient calculables facilement.

Par la suite, dans le chapitre 3 nous nous sommes efforcés de répondre à la question de l'existence et du calcul des équilibres des cas d'études du jeu de placement. Nous avons montré que pour la version générale du jeu de placement déterminer l'existence d'équilibre est NP-Complet. Néanmoins, en faisant une hypothèse restrictive sur ce jeu, nous obtenons un jeu de potentiel, que nous avons appelé *jeu de placement symétrique*, dans lequel l'existence d'équilibre de Nash est garantie. Nous avons montré que ce type de jeu est bien un jeu de potentiel et nous sommes intéressés à la forme de ses équilibres et à la complexité liée à leur calcul. En effet, prouver l'existence d'équilibres dans un jeu ne nous garantit pas que ceux-ci soient facilement calculables. Nous avons pu caractériser les équilibres dans un cas particulier de jeu de placement et montré que le cas général, considérant l'hypothèse restrictive sur ce jeu, était PLS-complet. Cela nous a amenés à considérer une variante non pondérée du jeu pour lequel nous avons prouvé qu'il existe des équilibres sous forme de stable maximal et donc calculables en temps polynomial. Nous avons également pu prouver que le stable maximum est le meilleur équilibre pour ce type de jeu.

Dans la deuxième partie de ce document, nous nous sommes intéressés à d'autres jeux de potentiel et au calcul distribué des équilibres en utilisant des algorithmes d'apprentissage et de minimisation de regret. Le chapitre 4 introduit la notion d'apprentissage et diverses dynamiques que sont la dynamique de la meilleure réponse, la dynamique de réplication, ainsi qu'un algorithme de minimisation de regret associé aux problèmes de Multi-Armed Bandit.

Dans le chapitre 5, nous nous sommes focalisés sur un jeu de placement basé sur un problème issu de la théorie des graphes : Max-Cut. Le problème Max-Cut consiste à diviser en deux parties l'ensemble des sommets d'un graphe donné, de façon à maximiser la somme des poids des arêtes traversant la partition. En interprétant ces coupes comme les équilibres de Nash purs d'un jeu de potentiel, nous avons pu proposer un algorithme probabiliste distribué convergeant vers ceux-ci. Nous avons pu prouver, grâce à une fonction de potentiel, qu'en appliquant notre algorithme sur un graphe complet de n sommets, une coupe localement maximale est atteinte en $O(\log \log n)$ étapes. En ce qui concerne un graphe quelconque, nous avons prouvé qu'une coupe localement maximale est atteinte en moyenne en $4\Delta|E|$ étapes, avec Δ le degré maximum du graphe.

Nous avons finalement étendu nos résultats à une application pour la gestion d'interférences dans des réseaux mobiles. Nous avons pu, pour les différents jeux, mettre en place des algorithmes afin de converger vers des états d'équilibre et prouver cette convergence.

Cependant, ces différents jeux de potentiel ne nous ont pas permis de comparer les approches d'apprentissage et de minimisation de regret. En effet lors de l'étude du problème Max-Cut nous nous sommes intéressés aux techniques de minimisation de regret, néanmoins les résultats n'étaient pas concluants, étant donnée la taille réduite de l'ensemble des stratégies possibles. Pour cela, nous nous sommes focalisés, dans le chapitre 6, sur un autre système en concurrence : la prédiction de violation de qualité de service dans un problème de sélection d'offres de qualité de service parmi divers fournisseurs d'accès, où nous comparons algorithmes d'apprentissage et minimisation de regret. Nous avons fourni un modèle pour le problème rencontré par un client au moment de choisir une offre de qualité de service pouvant rencontrer des échecs, dans un contexte où la fiabilité des fournisseurs peut changer. Ce problème est équivalent à celui d'un joueur ayant une utilité variable face à des machines dynamiques dans un problème de Multi-Armed Bandit. Nous avons proposé une adaptation d'un algorithme de Multi-Armed Bandit et de la dynamique de réplication permettant d'apprendre la fiabilité des fournisseurs d'accès.

Travaux futurs et perspectives. Dans ce document, nous avons présenté plusieurs variantes de notre jeu de placement obtenues par des hypothèses restrictives sur le jeu dans sa version générale. Néanmoins, nous pourrions nous intéresser à un cas encore plus général. En effet, dans les jeux d'ordonnancement les ressources considérées ne sont pas toujours uniformes. À chaque ressource est alors associée une fonction de coût dépendant de la vitesse de celle-ci.

La première question à se poser serait de savoir sous quelles conditions les résultats obtenus, pour les différentes variantes du jeu, restent valides (existence d'équilibre, complexité, etc.) si nous prenons en considération la vitesse des ressources.

Une autre question abordable serait de compléter les différentes variantes et caractérisations d'équilibres et notamment la qualité des équilibres. En effet, nous avons pu montrer que pour le jeu symétrique non pondéré le meilleur équilibre est le stable maximum du graphe associé au jeu, ce qui permet de mesurer le prix de l'anarchie. La question du meilleur équilibre ne se pose pas dans le cas du jeu de placement asymétrique, car l'existence d'équilibre y est incertaine, mais qu'en est-il du meilleur équilibre dans le jeu symétrique ?

Enfin, nous pourrions nous intéresser à l'apprentissage d'équilibres dans le jeu de placement symétrique. Une première approche serait de considérer les algorithmes distribués probabilistes que nous avons développés dans le chapitre 5. Ces algorithmes découlants d'une analyse de la méthode de Berenbrick *et al.* sur l'ordonnancement, nous pouvons nous poser la question de son efficacité sur notre jeu de placement qui en est une généralisation. Lors de l'étude sur le problème de Max-Cut, nous avons pu remarquer le souci de l'impact sur l'utilité du voisinage lors du déplacement d'un unique sommet, ainsi que de l'action simultanée des différents joueurs. Cette analyse nous laisse à penser que nous risquons de faire face au même problème dans le jeu de placement dans lequel la notion de voisinage est d'autant plus présente. Néanmoins pour le problème de Max-Cut nous avons réussi à contourner cela en déterminant une probabilité de déplacement fixe permettant une convergence vers un équilibre de Nash, ce qui nous conforte dans l'idée qu'une telle probabilité peut être obtenue pour le jeu de placement.

Une seconde approche pour l'apprentissage serait de considérer des algorithmes d'apprentissage tels que la dynamique de réplication ou des algorithmes de minimisation de regret dont nous avons fait l'étude dans le chapitre 6.

Concernant l'étude de la fiabilité des NSPs établie dans le chapitre 6, nous avons pu établir que le comportement du NSP, lors de la sélection d'une offre satisfaisant le client, a un impact sur les choix de ce dernier et donc sur l'ensemble du système. L'offre proposée par un NSP étant composée d'un chemin interne ainsi que d'un chemin externe menant à la destination, ce NSP peut alors être considéré comme un joueur devant choisir une offre de QoS correspondant à ce chemin externe. Chaque NSP effectuerait alors un apprentissage menant à des jeux en cascade. Nous pouvons alors nous poser la question de l'amélioration de la fiabilité des NSPs si chacun d'eux effectue un apprentissage.

Partie III

Annexe

Réallocation dynamique de tâches

Dans ce qui suit, nous sommes intéressés au problème de la réallocation dynamique de tâches. Nous présentons quelques résultats tirés des travaux de Petra Berenbrink sur l'ordonnancement, notamment [Berenbrink *et al.*, 2012].

Nous nous intéressons au problème de la réallocation dynamique des tâches de manière distribuée, c'est-à-dire un équilibrage de charge étant donné un placement initial aléatoire des différentes tâches sur les ressources. Nous disposons de n tâches auxquelles est associé un poids $w_i \geq 1$ pour tout $i \in [1, n]$, et m ressources uniformes. Le modèle est celui présenté au chapitre 1, il s'agit un cas spécial des jeux de congestion.

L'objectif du protocole étudié est d'obtenir un équilibrage de charges équitable sur les m ressources de manière distribuée. Le protocole utilisé par chaque tâche afin d'optimiser son utilité est identique, aucune tâche ne peut être identifiée par rapport à une autre, nous sommes dans un système distribué anonyme.

À chaque étape, chacune des tâches effectue le protocole suivant de manière simultanée. La tâche choisit une ressource au hasard parmi les m ressources. Elle compare alors la charge de sa ressource actuelle avec celle de la ressource sélectionnée aléatoirement, celle-ci pouvant être la ressource à laquelle la tâche est actuellement allouée. Si la ressource tirée aléatoirement possède une charge plus faible la tâche migre vers cette ressource avec une probabilité dépendant de la différence entre les charges des deux ressources, plus celle-ci est intéressante, plus la tâche a une probabilité forte de migrer vers la ressource sélectionnée.

L'intérêt de la méthode établie dans [Berenbrink *et al.*, 2012] est le temps nécessaire au système afin de converger vers un équilibre de Nash, les résultats obtenus étant une convergence en $O(\log(n) + m * \log(m))$ dans le cas des tâches non pondérées et $O(nm\Delta^5)$ dans le cas de poids entiers sur les tâches avec Δ le poids maximum de celles-ci. L'algorithme de Petra Berenbrink converge vers un ε -équilibre de Nash dans le cas de tâches pondérées en $O(nm\Delta^3\varepsilon^{-2})$.

Nous considérons par la suite que le nombre de tâches à réallouer est au moins égal au nombre de ressources disponibles. Chaque tâche i possède un poids w_i et nous notons Δ la valeur maximum parmi les poids de toutes les tâches, soit $\Delta = \max_{i \in [1, n]} \{w_i\}$ et N le poids total de ces tâches avec $N = \sum_{i=1}^n w_i$. À chaque étape, les tâches sont affectées à une et une seule ressource, une tâche ne pouvant être partitionnée et traitée par des ressources différentes. La charge moyenne d'une ressource est notée $\bar{x}$ et vaut $\frac{1}{m} \sum_{j=1}^m (x_j)$, soit $\frac{N}{m}$.

La ressource à laquelle une tâche i est allouée à l'étape courante est notée r_i pour tout $i \in [1, n]$. L'assignement des n tâches aux ressources est représenté par un vecteur de taille m , $X(t) = (X_1(t), \dots, X_m(t))$ où chaque composante $X_j(t)$ représente la charge de la ressource $j \in [1, m]$ à la fin de l'étape t , c'est-à-dire la somme des poids des tâches allouées à cette ressource.

7.1

Réallocation de tâches pondérées

Soit $X(0) = (X_1(0), \dots, X_n(0))$ l'état initial du système qui découle d'un assignement aléatoire des tâches aux ressources. Le protocole suivant, exécuté par les différentes tâches à chaque étape t , nous donne la transition d'un état $X(t)$ vers $X(t+1)$, avec $0 \leq \varepsilon \leq 1$ et $\rho = \frac{\varepsilon}{8}$, appelé facteur de ralentissement.

Algorithme 7.1 – Protocole de réallocation de tâches pondérées [Berenbrink et al., 2012]

Données : Un assignement aléatoire des tâches aux ressources

Pour chaque étape t **faire**

Pour chaque tâche i **en parallèle faire**

 Choisir de manière aléatoire uniforme une ressource j

Si $X_{r_i}(t) \geq X_j(t) + (1 + \varepsilon)w_i$ **Alors**

 Aller en j avec la probabilité $\rho \left(1 - \frac{X_j(t)}{X_{r_i}(t)}\right)$

Fin

Fin

Fin

La preuve du temps de convergence de cet algorithme passe par l'étude d'une fonction de potentiel, en bornant la perte moyenne de cette fonction de potentiel en une étape. En effet, la fonction de potentiel ayant un minimum correspondant à un équilibre de Nash montre que celle-ci décroît en moyen

d'un facteur donné nous permet de borner le temps moyen de convergence vers cet équilibre.



Théorème 7.1 – Temps de convergence vers un ε -équilibre de Nash [Berenbrink *et al.*, 2012]

Soit $0 \leq \varepsilon \leq 1$, $\rho = \frac{\varepsilon}{8}$ et $\Delta \geq 1$ le poids maximum des tâches. Soit T le nombre de rounds nécessaires à l'algorithme 7.1 pour converger vers un ε -équilibre de Nash pour la première fois. Alors

$$E[T] = O(nm\Delta^3\varepsilon^{-2})$$

7.2

Réallocation de tâches non pondérées

Dans le cas de tâches non pondérées, le modèle reste le même avec $\varepsilon = 1$ et donc $\rho = \frac{1}{8}$. Nous avons n tâches identiques de poids unitaire à assigner sur m ressources uniformes. Une tâche choisit une ressource au hasard parmi les m et si la charge de cette ressource est inférieure d'au moins 2 par rapport à sa ressource actuelle alors la tâche migre avec une probabilité $\rho \left(1 - \frac{\text{nouvelle charge}}{\text{ancienne charge}}\right)$ sachant que le facteur de ralentissement est $\rho = \frac{1}{8}$. Le protocole devient le suivant.

Algorithme 7.2 – Protocole de réallocation de tâches non pondérées [[Berenbrink *et al.*, 2012]]

Données : Un assignement aléatoire des tâches aux ressources

Pour chaque étape t **faire**

Pour chaque tâche i **en parallèle faire**

 Choisir de manière aléatoire uniforme une ressource j

Si $X_{r_i}(t) \geq X_j(t) + 1$ **Alors**

 Aller en j avec la probabilité $\rho \left(1 - \frac{X_j(t)}{X_{r_i}(t)}\right)$

Fin

Fin

Fin


Théorème 7.2 – Temps de convergence vers un équilibre de Nash [Berenbrink et al., 2012]

Soit $X(0) = (X_1(0), \dots, X_n(0))$ une configuration initiale. Soit T le nombre d'étapes nécessaires à l'algorithme 7.2 pour atteindre un équilibre de Nash pour la première fois, nous avons

$$E[T] = O(\log(n) + m \log(m))$$

De plus $T = O(\log(n) + m \log(m))$ avec une probabilité au moins $1 - \frac{1}{m}$.

Preuve : De même que pour la version pondérée, la preuve de ce temps de convergence passe par l'étude d'une fonction de potentiel. Afin d'obtenir un temps de convergence il est nécessaire de borner la perte moyenne de la fonction de potentiel en une étape, ce qui nous donne qu'à chaque étape la fonction de potentiel décroît d'au moins un facteur $\frac{1}{32}$ si $\Phi(X(t)) > n$, et d'un facteur $\frac{1}{8m}$ sinon. La fonction de potentiel utilisée est la suivante :

$$\Phi(x) = \sum_{j=1}^m (x_j - \bar{x})^2$$

Cette fonction atteint un minimum pour $x_j = \bar{x}$, $\forall j \in m$ où $\Phi(x) = 0$, sachant que $\bar{x} = \frac{N}{m}$ correspondant à la charge moyenne d'une ressource. Elle est équivalente à la fonction de potentiel de Rosenthal [Rosenthal, 1973]. Cette fonction est bien une fonction de potentiel. En effet, la modification de l'utilité d'une tâche entraînée par sa migration vers une nouvelle ressource vaut exactement la moitié de la variation de Φ . L'objectif est de montrer qu'en moyenne la valeur de $\Phi(x)$ diminue à chaque étape. Étant donné $X(t)$ nous pouvons déterminer $E[\Phi(X(t+1))|X(t)]$ la perte moyenne de la fonction de potentiel après une étape.

$$\begin{aligned} E[\Phi(X(t+1))|X(t)] &= E\left[\sum_{j=1}^m (X_j(t+1) - \bar{x})^2 | X(t) = x\right] \\ &= \sum_{j=1}^m (E[(X_j(t+1)) | X(t) = x] - \bar{x})^2 \\ &\quad + \sum_{j=1}^m V[(X_j(t+1)) | X(t) = x] \end{aligned}$$

Pour borner la différence de potentiel durant un temps T fixé, il faut :

1. borner $\sum_{j=1}^m (E[(X_j(t+1))|X(t) = x] - \bar{x})^2$
2. borner $\sum_{j=1}^m V[(X_j(t+1))|X(t) = x]$

L'espérance et la variance ont été bornées dans le cas de tâches pondérées. Appliqué au cas non pondéré les inégalités suivantes ont été obtenues.

$$\begin{aligned} \sum_{j=1}^m V[(X_j(t+1))|X(t) = x] &< (2 - \varepsilon) \sum_{j=1}^m \sum_{k=j+1}^m E[W_{jk}](x_j - x_k) \\ \sum_{j=1}^m (E[(X_j(t+1))|X(t) = x] - \bar{x})^2 &> \Phi(x) - \sum_{j=1}^m \sum_{k=j+1}^m 2E[W_{jk}](x_j - x_k) \\ \sum_{j=1}^m (E[(X_j(t+1))|X(t) = x] - \bar{x})^2 &< \Phi(x) - (2 - 4\rho) \sum_{j=1}^m \sum_{k=j+1}^m (x_j - x_k) l_{j,k} \end{aligned}$$

$E[W_{jk}]$ est le poids total moyen des ressources migrant de j à k .

Si $x_j - x_k \geq 2$ alors $E[W_{jk}] = \frac{\rho(x_j - x_k)}{m}$.

Sinon $E[W_{jk}] = 0$.

Grâce à ces bornes les auteurs ont montré une borne supérieure de $E[\Phi(X(t+1))|X(t)]$. En effet

$$\begin{aligned} E[(X_j(t+1))|X(t) = x] &= \sum_{j=1}^m (E[(X_j(t+1))|X(t) = x] - \bar{x})^2 \\ &+ \sum_{j=1}^m V[(X_j(t+1))|X(t) = x] \\ &< \Phi(x) - \left(\frac{1}{2}\right) \sum_{j=1}^m \sum_{k=j+1}^m E[W_{jk}](x_j - x_k) \end{aligned}$$

Cela a permis d'arriver à borner la perte moyenne à chaque étape de la fonction de potentiel pour n'importe quelle étape $t > 0$.

- $E[(X_j(t+1))] \leq \max\{m, (1 - \frac{1}{32})E[\Phi(X(t))]\}$
- $E[(X_j(t+1))] \leq (1 - \frac{1}{8m})E[\Phi(X(t))]$

De ce fait, le temps de convergence a pu être établi. □

Table des figures

1.1	Représentation d'un jeu sous forme extensive	12
1.2	Exemple d'équilibre de Nash dans un jeu d'ordonnancement	19
1.3	Meilleure réponse amenant à un équilibre	20
2.1	Exemple d'ordonnancement et son équivalent sous forme de jeu de placement	26
2.2	Résumé du contexte atomique	27
2.3	Représentation du jeu dans l'exemple 1	30
2.4	Représentation du jeu dans l'exemple 2	32
2.5	Exemple d'équilibre de Nash dans un jeu de placement symétrique $x_{v_1} = \frac{n}{3}$, $x_{v_2} = \frac{n}{3}$ et $x_{v_3} = \frac{n}{3}$	34
2.6	Exemple d'équilibre de Nash dans un jeu de placement symétrique non pondéré $x_{v_1} = \frac{n}{3}$, $x_{v_2} = \frac{n}{3}$ et $x_{v_3} = \frac{n}{3}$	35
2.7	Exemple d'équilibre de Nash dans un jeu de placement symétrique non pondéré $x_{v_1} = \frac{n}{2}$ et $x_{v_3} = \frac{n}{2}$	35
3.1	Illustration d'une partie du jeu correspondant à la clause $cl_j = (u_1, \overline{u_2}, u_3)$	43
3.2	Trois types d'équilibres pour une clause $cl_j = (u_a, u_b, u_c)$	46
3.3	Exemple d'assignation de E3-SAT vers un NE pur	48
3.4	Construction du graphe G' à partir de G	56
3.5	Équilibres non atomiques sur une chaîne pondérée de m sommets ($\forall(i, j) \in E, d_{i,j} = \frac{1}{2}$). La dernière chaîne représente un équilibre ne dépendant pas de la parité de m . Les sommets en noirs sont les sommets chargés, ceux en blanc sont vides.	60
3.6	$\Phi = 9$ Équilibre local	73
3.7	$\Phi = 9$ Déplacement de charge	73
3.8	$\Phi = 7$ Meilleur équilibre	73
5.1	Exemple de graphe biparti non pondéré	94
5.2	Différence entre coupe localement maximale et coupe optimale.	94
5.3	Configuration n'étant pas une coupe localement maximale	96
5.4	Meilleures réponses depuis la configuration 5.3	96
5.5	Temps moyen de convergence pour parvenir à un équilibre dans des graphes aléatoires	105
5.6	Temps moyen de convergence pour parvenir à un équilibre dans des chemins et cycles	106
5.7	Interférence inter cellule	107
5.8	Interprétation sous forme de graphe représentant une arête du problème de Max-Cut	108
5.9	Station de base hexagonale et son voisinage	109
5.10	Temps de convergence en fonction du nombre de RBs	121
5.11	Vitesse de convergence en fonction de la charge	122

6.1	Un client réseau est relié à 3 NSPs. Chaque NSP i dispose d'un ensemble d'offres de QoS, Q_k , ainsi que d'un ensemble de chemins disjoints pour atteindre un préfixe ciblé, $\Pi^{(k)}$. Lorsque le client c envoie une requête notée q_c , chaque NSP k peut effectuer une proposition d'offre q_k à un prix $g_k(.)$	125
6.2	Scénario de la négociation	126
6.3	Problème sous forme de Multi-Armed Bandit	130
6.4	Comparaison des algorithmes UCB dans le scénario 1 - Le regret du client	139
6.5	Récompense obtenue par le client	140
6.6	Pourcentage de requêtes ayant échoué	140

Liste des tableaux

1.1	Représentation d'un jeu sous forme matricielle	11
2.1	Index des notations relatives au jeu de placement (G, D, n) .	37
6.1	Les offres des NSPs choisies	137
6.2	Politique de proposition d'offre des NSPs	138
6.3	Les choix du client et les échecs des NSPs - Simulations avec violation sur les chemins externes et la capacité restante . .	141
6.4	Les choix du client et les échecs des NSPs - Simulations avec violation sur les chemins externes uniquement	141

Bibliographie

- [DBL, 2013] (2013). *Proceedings of the 9th International Conference on Network and Service Management, CNSM 2013, Zurich, Switzerland, October 14-18, 2013*. IEEE.
- [DBL, 2014] (2014). *IEEE Wireless Communications and Networking Conference, WCNC 2014, Istanbul, Turkey, April 6-9, 2014*. IEEE.
- [LTE, 2008] (June 2008). 3gpp ts 36.211 v8.3.0, evolved universal terrestrial radio access; physical channels and modulation. Rapport technique.
- [Adolphs et Berenbrink, 2011] Adolphs, C. P. J. et Berenbrink, P. (2011). Distributed selfish load balancing with weights and speeds. *CoRR*, abs/1109.6925.
- [Adolphs et Berenbrink, 2012] Adolphs, C. P. J. et Berenbrink, P. (2012). Distributed selfish load balancing with weights and speeds. In [Kowalski et Panconesi, 2012], pages 135–144.
- [Adouane et al., 2014a] Adouane, A., Rodier, L., Khawam, K., Cohen, J. et Tohme, S. (2014a). Distributed load balancing game for inter-cell interference coordination. In *European Wireless 2014; 20th European Wireless Conference; Proceedings of*, pages 1–6.
- [Adouane et al., 2014b] Adouane, A., Rodier, L., Khawam, K., Cohen, J. et Tohmé, S. (2014b). Game theoretic framework for inter-cell interference coordination. In [DBL, 2014], pages 57–62.
- [Amigo et al., 2012] Amigo, I., Vaton, S., Chonavel, T. et Larroca, F. (2012). Maximum delay computation for interdomain path selection. *Int. Journal of Network Management*, 22(2):162–179.
- [Apt et al., 2014] Apt, K. R., Rahn, M., Schäfer, G. et Simon, S. (2014). *Coordination Games on Graphs (Extended Abstract)*, pages 441–446. Springer International Publishing, Cham.
- [Assaad, 2008] Assaad, M. (2008). Optimal fractional frequency reuse (ffr) in multicellular ofdma system. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5.
- [Audibert et Bubeck, 2009] Audibert, J.-Y. et Bubeck, S. (2009). Minimax policies for adversarial and stochastic bandits. In *COLT*.
- [Auer et al., 2002] Auer, P., Cesa-Bianchi, N. et Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256.
- [Auer et al., 2003] Auer, P., Cesa-Bianchi, N., Freund, Y. et Schapire, R. E. (2003). The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77.

- [Auger *et al.*, 2013] Auger, D., Cohen, J., Coucheney, P. et Rodier, L. (2013). Distributed selfish algorithms for the max-cut game. *In* [Gelenbe et Lent, 2013], pages 45–54.
- [Barahona, 1980] Barahona, F. (1980). *On the Complexity of Max Cut*. RR // Université Scientifique et Médicale et Institut National de Polytechnique de Grenoble, Mathématiques Appliquées et Informatique. Univ.
- [Becvar *et al.*, 2012] Becvar, Z., Bestak, R. et Kencl, L., éditeurs (2012). *NETWORKING 2012 Workshops - International IFIP TC 6 Workshops, ETICS, HetsNets, and CompNets, Held at NETWORKING 2012, Prague, Czech Republic, May 25, 2012. Proceedings*, volume 7291 de *Lecture Notes in Computer Science*. Springer.
- [Berenbrink *et al.*, 2012] Berenbrink, P., Friedetzky, T., Hajirasouliha, I. et Hu, Z. (2012). Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks. *Algorithmica*, 62(3-4):767–786.
- [Berge, 1958] Berge, C. (1958). *Théorie des graphes et ses applications*. Collection Univesitaire des Mathématiques, Dunod, Paris.
- [Berger, 2003] Berger, U. (2003). Fictitious play in $2 \times n$ games. *Game theory and information*, EconWPA.
- [Boudreau *et al.*, 2009] Boudreau, G., Panicker, J., Guo, N., Chang, R., Wang, N. et Vrzic, S. (2009). Interference coordination and cancellation for 4G networks. *IEEE Communications Magazine*, 47:74–81.
- [Brouwer,] Brouwer, L. E. J. Über abbildung von mannigfaltigkeiten. *Mathematische Annalen*, 71(1):97–115.
- [Christodoulou *et al.*, 2012] Christodoulou, G., Mirrokni, V. S. et Sidiropoulos, A. (2012). Convergence and approximation in potential games. *Theor. Comput. Sci.*, 438:13–27.
- [Correa et Stier-Moses, 2011] Correa, J. R. et Stier-Moses, N. E. (2011). Wardrop equilibria. *Wiley Encyclopedia of Operations Research and Management Science*.
- [Cournot, 1838] Cournot, A. (1838). *Recherches sur les principes mathématiques de la théorie des richesses*. L. Hachette.
- [Daskalakis *et al.*, 2006] Daskalakis, C., Goldberg, P. W. et Papadimitriou, C. H. (2006). The complexity of computing a nash equilibrium. pages 71–78. ACM Press.
- [Delorme et Poljak, 1990] Delorme, C. et Poljak, S. (1990). *Laplacian Eigenvalues and the Maximum Cut Problem*. Rapports de recherche. Université de Paris-Sud, Centre d’Orsay, Laboratoire de Recherche en Informatique.
- [Diekmann *et al.*, 1999] Diekmann, R., Frommer, A. et Monien, B. (1999). Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25(7):789 – 812.

- [Douvillie *et al.*, 2008] Douville, R., Roux, J. L., Rougier, J. et Secci, S. (2008). A service plane over the PCE architecture for automatic multi-domain connection-oriented services. *IEEE Communication Magazine*.
- [Ellenbeck *et al.*, 2008] Ellenbeck, J., Hartmann, C. et Berlemann, L. (2008). Decentralized inter-cell interference coordination by autonomous spectral reuse decisions. In *Wireless Conference, 2008. EW 2008. 14th European*, pages 1–7.
- [Even-Dar *et al.*, 2007] Even-Dar, E., Kesselman, A. et Mansour, Y. (2007). Convergence time to nash equilibrium in load balancing. *ACM Transactions on Algorithms*, 3(3).
- [Fabrikant *et al.*, 2004] Fabrikant, A., Papadimitriou, C. H. et Talwar, K. (2004). The complexity of pure nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 604–612.
- [Freund et Schapire, 1995] Freund, Y. et Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory, Second European Conference, EuroCOLT '95, Barcelona, Spain, March 13-15, 1995, Proceedings*, pages 23–37.
- [Garey et Johnson, 1979] Garey, M. R. et Johnson, D. S. (1979). *Computers and intractability : a guide to the theory of NP-completeness*. W. H. Freeman, first edition édition.
- [Gelenbe et Lent, 2013] Gelenbe, E. et Lent, R., éditeurs (2013). *Information Sciences and Systems 2013 - Proceedings of the 28th International Symposium on Computer and Information Sciences, ISCIS 2013, Paris, France, October 28-29, 2013*, volume 264 de *Lecture Notes in Electrical Engineering*. Springer.
- [Gelly *et al.*, 2006] Gelly, S., Wang, Y., Munos, R. et Teytaud, O. (2006). Modification of UCT with Patterns in Monte-Carlo Go. Rapport de recherche RR-6062, INRIA.
- [Gesbert *et al.*, 2007] Gesbert, D., Oien, G. E., Kiani, S. G. et Gjendemo, A. (2007). Adaptation, coordination and distributed resource allocation in interference-limited wireless networks. *Proceedings of the IEEE, Volume 95, N°12, December 2007*.
- [Gittins et Gittins, 1979] Gittins, A. J. C. et Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, pages 148–177.
- [Goemans et Williamson, 1995] Goemans, M. X. et Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145.

- [Goldberg, 2004] Goldberg, P. W. (2004). Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 131–140, New York, NY, USA. ACM.
- [Gourvès et Monnot, 2010] Gourvès, L. et Monnot, J. (2010). The max k-cut game and its strong equilibria. In Kratochvíl, J., Li, A., Fiala, J. et Kolman, P., éditeurs : *TAMC*, volume 6108 de *Lecture Notes in Computer Science*, pages 234–246. Springer.
- [Gradinariu et Tixeuil, 2006] Gradinariu, M. et Tixeuil, S. (2006). Conflict Managers for Self-stabilization without Fairness Assumption. Rapport technique 1459, LIP6.
- [Grigoriadis et Khachiyan, 1995] Grigoriadis, M. D. et Khachiyan, L. G. (1995). A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2).
- [Jain, 1991] Jain, R. (1991). *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*. John Wiley & Sons.
- [Johnson et al., 1988] Johnson, D. S., Papadimitriou, C. H. et Yannakakis, M. (1988). How easy is local search? *Journal of Computer and System Sciences*, 37(1):79 – 100.
- [Karp, 1972] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. et Thatcher, J., éditeurs : *Complexity of Computer Computations*, pages 85–103. Plenum Press.
- [Kearns et al., 2001] Kearns, M. J., Littman, M. L. et Singh, S. P. (2001). Graphical models for game theory. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 253–260, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Kernighan et Lin, 1970] Kernighan, B. W. et Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49:291–307.
- [Khot et al., 2004] Khot, S., Kindler, G. et Mossel, E. (2004). Optimal inapproximability results for max-cut and other 2-variable csps. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 146–154.
- [Kleinberg et Tardos, 2005] Kleinberg, J. et Tardos, E. (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [Kocsis et Szepesvári, 2006] Kocsis, L. et Szepesvári, C. (2006). Bandit based monte-carlo planning. In *17th European Conference on Machine Learning, Berlin, Germany, Proceedings*, volume 4212 de *Lecture Notes in Computer Science*, pages 282–293.

- [Koutsoupias et Papadimitriou, 1999] Koutsoupias, E. et Papadimitriou, C. (1999). Worst-case equilibria. In *IN PROCEEDINGS OF THE 16TH ANNUAL SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE*, pages 404–413.
- [Kowalski et Panconesi, 2012] Kowalski, D. et Panconesi, A., éditeurs (2012). *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*. ACM.
- [Kuhn et Tucker, 1951] Kuhn, H. W. et Tucker, A. W. (1951). Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif. University of California Press.
- [Lai et Robbins, 1985] Lai, T. L. et Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, pages 4–22.
- [Lamali et al., 2012] Lamali, M. L., Barth, D. et Cohen, J. (2012). Reputation-aware learning for sla negotiation. In *Networking Workshops*, pages 80–88.
- [Lovasz, 1994] Lovasz, L. (1994). Stable sets and polynomials. *Discrete Mathematics*, 124(1-3):137 – 153.
- [Mao et al., 2008] Mao, X., Maaref, A. et Teo, K. H. (2008). Adaptive soft frequency reuse for inter-cell interference coordination in sc-fdma based 3gpp lte uplinks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–6.
- [Monderer et Shapley, 1996a] Monderer, D. et Shapley, L. S. (1996a). Fictitious play property for games with identical interests. *Journal of Economic Theory*, 68(1):258–265.
- [Monderer et Shapley, 1996b] Monderer, D. et Shapley, L. S. (1996b). Potential games. *Games and Economic Behavior*, 14(1):124 – 143.
- [Nash, 1951] Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2):286–295.
- [Nash, 1950] Nash, J. F. (1950). Equilibrium points in n -person games. *Proc. of the National Academy of Sciences*, 36:48–49.
- [Neumann et Morgenstern, 1944] Neumann, J. V. et Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.
- [Nisan et al., 2007] Nisan, N., Roughgarden, T., Tardos, E. et Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA.
- [Ottens et al., 2012] Ottens, B., Dimitrakakis, C. et Faltings, B. (2012). Duct : An UCB approach to distributed constraint optimization problems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

- [Papadimitriou, 1994] Papadimitriou, C. H. (1994). On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498 – 532.
- [Quek et al., 2009] Quek, T., Lei, Z. et Sun, S. (2009). Adaptive interference coordination in multi-cell ofdma systems. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pages 2380–2384.
- [Rahman et al., 2009] Rahman, M., Yanikomeroglu, H. et Wong, W. (2009). Interference avoidance with dynamic inter-cell coordination for downlink lte system. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6.
- [Robbins, 1952] Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.
- [Rodier et al., 2013] Rodier, L., Auger, D., Cohen, J. et Pouyllau, H. (2013). SLA learning from past failures, a multi-armed bandit approach. In [DBL, 2013], pages 277–283.
- [Rosenthal, 1973] Rosenthal, R. W. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67.
- [Sastry et al., 1994] Sastry, P. S., Phansalkar, V. V. et Thathachar, M. A. L. (1994). Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information. *IEEE Trans Systems, Man, and Cybernetics*, 24.
- [Schaffer, 1991] Schaffer, A. A. (1991). Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87.
- [Shapley, 1953] Shapley, L. S. (1953). A value for n-person games. *Contributions to the theory of games*, 2:307–317.
- [Sklar, 1997] Sklar, B. (1997). Rayleigh fading channels in mobile digital communication systems .i. characterization. *Comm. Mag.*, 35(7):90–100.
- [Wang et Crowcroft, 1996] Wang, Z. et Crowcroft, J. (1996). Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234.
- [Xiao et Boutaba, 2005] Xiao, J. et Boutaba, R. (2005). Qos-aware service composition and adaptation in autonomic communication. *IEEE Journal on Selected Areas in Communications*, 23:2344–2360.

Titre : Existence et calcul distribué d'équilibres dans des jeux de congestion généralisés

Mots clés : théorie des jeux, équilibre de Nash, jeux de potentiel, apprentissage d'équilibre

Résumé : Cette thèse se focalise sur les jeux de potentiel et une généralisation d'un jeu d'ordonnement dans un graphe que nous avons appelé jeu de placement. Dans ce jeu, le coût d'un joueur est impacté par son voisinage. Nous pouvons illustrer cela avec un exemple : le placement de joueurs dans un train, pour lesquels la présence de voisins directs influe sur le bien-être.

Les résultats de cette thèse se divisent en deux parties.

Tout d'abord, nous étudions ces jeux en considérant l'existence et les propriétés de structure des équilibres. Nous nous posons la question fondamentale de savoir s'il existe des équilibres de Nash dans le jeu de placement. Si tel est le cas, nous tâchons de déterminer si ces équilibres sont facilement calculables. Dans le cas où il n'existe pas d'équilibre nous prouvons la

NP-complétude du problème.

Dans un second temps nous nous intéressons à la notion de calcul distribué d'équilibre de Nash dans des jeux de placement. En particulier nous considérons un jeu basé sur le problème de Max-Cut, qui a été plus étudié en théorie des graphes. Cela nous a permis d'étendre nos travaux à une application aux réseaux mobiles pour la gestion d'interférences dans les réseaux sans fils. Nous avons pu, pour les différents jeux, mettre en place des algorithmes distribués de calcul d'équilibres et étudier leur convergence. Parallèlement, nous avons étendu les travaux de Max-Cut à un problème de sélection d'offre de qualité de service parmi divers fournisseurs d'accès. Nous comparons les performances d'algorithmes de calcul distribué d'équilibres et de minimisation de regret.

Title : Existence and distributed computation of equilibria in generalized congestion games

Keywords : game theory, Nash equilibria, potential games, learning equilibria

Abstract : This thesis focuses on potential games and a generalized load balancing game in a graph we called placement game. In this game, the cost of a player is affected by its neighbors. We can illustrate this with an example : the placement of players on a train, where the presence of direct neighbors affects their well-being.

The results of this thesis are divided into two parts.

First, we study these games considering the existence and structural properties of equilibria. We ask ourselves the fundamental question of whether there are Nash equilibria in the placement game. If this is the case we aim to determine if they are easily calculable, if there is no such equilibria we

prove the NP-completeness of the problem.

Secondly we focus on the concept of distributed algorithms to compute Nash equilibria in placement games. In particular we consider a game based on the Max-Cut problem, which has been more frequently studied. This allowed us to expand our work to a mobile network application for managing interference in wireless networks. We were able, for those different games, to implement distributed algorithms to compute equilibria and study their convergence. Meanwhile, we have expanded the Max-Cut works with a selection of QoS offers problem from various network providers. We compare the performance of distributed algorithms and regret minimization.