République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme de Master

Spécialité : *Modélisation Mathématique et Techniques de Décision*

Thème

Calcul des Équilibres par la Programmation par Contraintes

Présenté par :

M^{elle}.SAOUDI Sylia

M^{elle}.Aguenana Rima

Devant le jury composé de :

Présidente :	**Mme KENDI Salima**	**MAA**	**U. A. Mira Béjaia.**
Rapporteur :	**Mme BOUCHAMA Kahina**	**MAB**	**U. A. Mira Béjaia.**
Examinatrice :	**Mme BOUIBED Karima**	**MCB**	**U. A. Mira Béjaia.**
Examinateur :	**Mr ZIANI Sofiane**	**MCB**	**U. A. Mira Béjaia.**

Promotion 2019-2020

Remerciements

Louange A Dieu, le miséricordieux, sans Lui rien de tout cela n'aurait pu être.

Au terme de ce travail nous exprimons nos plus sincères remerciements à notre promotrice ***Dr. Kahina Bouchama*** *pour l'honneur qu'elle nous a fait en acceptant de nous encadrer. Ses conseils précieux ont permis une bonne orientation dans la réalisation de ce modeste travail.*

N*ous tenons également à remercier les membres de jury pour l'honneur qu'ils nous ont fait en acceptant de juger ce travail, et d'avoir consacrer leurs temps pour sa lecture.*

N*ous tenons à adresser nos vifs remerciements à nos enseignements pour le savoir qui nous ont transmis durant notre cursus.*

E*nfin nous tenons à rendre hommage à toutes nos familles et nos amis pour le soutien qu'ils nous ont apportés durant toutes ces années d'études.*

Dédicaces

Je dédie ce modeste travail :

A celle qui m'a donnée vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère.

A mon père, école de mon enfance, qui a été mon ombre durant toutes les années d'études, Que dieu les gardes et les protège.

A mes chères sœurs Anissa, Siham et Wissam.

A mes chers frères Adel et Wassim.

A ma chère copine Sylia et toute sa famille.

A tous mes amis surtout mes proches et à tous les étudiants de ma promotion de département Recherche Opérationnelle de l'université de Bejaia.

A toute ma famille.

Rima Aguenana

Dédicaces

Je dédie ce modeste travail :

A mon exemple dans la vie, la lumière de mon chemin, à celui qui me donne la puissance de continuer, à mon chèr ***Père****.*

A ma mère, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, Que dieu les gardes et les protège .

A mes chères sœurs Yasmina et Thiziri et ma cousine Kanza pour leurs encouragements et vœux de tous les jours.

A mes chers frères Azzedine et Lounis.

A mes chèrs grand parents, que dieu les gardes pour nous.

A celle qui ma toujours soutenu, à ma chère copine Thiziri qui était toujours à mes cotés et à toute sa famille.

A ma chère Hayet Marzouk pour son aide et encouragement et à toute sa famille.

A ma chère Rima et toute sa famille.

A tous mes amis surtout mes proches et à tous les étudiants de ma promotion de département Recherche Opérationnelle de l'université de Bejaia.

A toute ma famille.

Sylia Saoudi

Table des matières

Introduction Générale **4**

1 Élements de la Théorie des Jeux **6**
1.1 Introduction 6
1.2 Concepts de base 6
1.3 Classification des jeux 8
1.4 Représentation d'un jeu 9
1.5 Concept de solution pour les jeux sous forme normale 13
1.5.1 Équilibre d'un jeu 14
1.6 Complexité du calcul des équilibres de Nash 15
1.7 Méthodes de calcul des équilibres de Nash 16
1.8 Conclusion 19

2 La Programmation par Contraintes (PPC) **20**
2.1 Introduction 20
2.2 Les problèmes de satisfaction de contraintes (CSP) 20
2.2.1 Définitions élémentaires 20
2.2.2 Représentation graphique d'un CSP 22
2.2.3 Le problème de satisfiabilité (SAT) : 23
2.2.4 La complexité des problèmes de satisfaction de contraintes 23
2.2.5 Modélisation des problèmes par des CSPs 24
2.3 Les méthodes de résolution des problèmes CSP 26
2.3.1 Générer et tester 26
2.3.2 Les méthodes par retours-arrière (backtrack) 27
2.3.3 Les techniques de filtrage et de propagation de contraintes 28
2.3.4 Le Forward-Checking 29
2.4 Les heuristiques de choix lors de recherche d'une solution 29
2.5 Conclusion 30

3 Applications de la PPC dans le calcul des équilibres de Nash **31**
3.1 Les jeux graphique et les CSPs 32
3.2 Les CP-nets et équilibre de Nash 34
3.3 Les jeux et les CSPs continu 35

3.4 Les jeux stratégiques et les softs CSP 36
3.5 Les jeux de contraintes 36
3.6 Conclusion 37

4 Modèle CSP pour la Représentation des Dynamiques de Recherche des Stratégies de Meilleures Réponses 38
4.1 Introduction 38
4.2 Position du problème 38
4.3 Le modèle proposé 39
4.4 Illustration du modèle proposé sur le jeu du dilemme du prisonnier 40
4.4.1 Principe du dilemme du prisonnier (DP) 40
4.4.2 Formulation classique du jeu du dilemme du prisonnier 41
4.4.3 Formulation du CSP associé au jeu du dilemme du prisonnier . . . 41
4.4.4 Résolution du CSP(P_D) 41
4.5 Illustration du modèle proposé sur un jeu sous forme graphique 43
4.5.1 Formulation du jeu (4.1) sous forme graphique 43
4.5.2 Formulation du CSP associé au jeu graphique (4.24) 44
4.5.3 Résolution du CSP(P^G) 47
4.6 Conclusion 50

Conclusion Générale 51

Bibliographie 52

Table des figures

1.1 Représentation d'un jeu sous forme extensive. 10
1.2 Représentation d'un jeu sous forme graphique. 11
1.3 Matrices locales de tous les joueurs de l'exemple 12

2.1 Graphe de contraintes du CSP P. 23
2.2 Problème de coloriage de carte géographique 25
2.3 Représentation du problème de coloriage de carte sous forme d'un graphe . 25

4.1 Représentation graphique du jeu (4.28). 45

Introduction Générale

La théorie des jeux est un outil mathématique qui étudie des situations où des individus (dits *joueurs*) sont conduits à faire des choix parmi un certain nombre d'actions possibles, et dans un cadre défini à l'avance (appelé *règles du jeu*). Les résultats de ces choix constituent une issue du jeu à laquelle est associé un gain pour chacun des participants. Ces résultats ne dépendent pas de la décision d'un seul joueur, mais également des décisions prises par d'autres participants. Cette théorie a été appliquée dans plusieurs domaines tels que la biologie, l'informatique, l'économie, etc.

L'équilibre de Nash est le concept de solution le plus connu dans la théorie des jeux non coopératifs et le plus étudié pour les jeux statiques dans lesquels chaque joueur est supposé connaître les stratégies de ses adversaires, et aucun joueur n'a intérêt à dévier seul de sa propre stratégie. Cependant, le calcul de cet équilibre est connu pour être un problème NP difficile, ce qui fait que la recherche de cet équilibre devient assez compliqué lorsque la taille du jeu devient importante. Ces dernières années, bien que pas nombreux, il existe certains travaux contemporains dans la littérature qui ont fait recours à la programmation par contraintes pour faire face à la complexité du calcul des équilibres de Nash.

La programmation par contraintes est un paradigme de programmation qui permet de traiter de nombreux problèmes combinatoires de nature décisionnelle. Apparue dans les années 1970, l'approche par programmation par contraintes consiste à modéliser ces problèmes en réseaux de contraintes (CN pour Constraint Networks) : ils sont définis par un ensemble de variables, représentant des inconnues, et un ensemble de contraintes représentant certaines limitations sur les valeurs de ces inconnues, le problème résultant est appelé problème de satisfaction de contraintes (CSP pour Constraint Satisfaction Problem).

Il existe plusieurs approches pour la résolution des problèmes CSP, ces approches peuvent être exploitées pour résoudre des problèmes de jeux vus les liens existants entre ces deux théorie, en particulier dans la recherche de l'équilibre de Nash. En effet, Bordeaux et Pajot [19] ont adopté l'approche par la programmation par contraintes pour le calculer, de même pour Krzysztof, Rossi et Venable [18] qui ont quant à eux comparé les notions d'optimalité dans les jeux stratégiques et les softs-CSP. Notons que la plupart des travaux effectués dans ce domaine se sont surtout focalisés sur le concept d'équilibre de Nash, vue son importance majeure en théorie des jeux[19, 16].

Par conséquent, notre but à travers ce mémoire est justement de présenter une synthèse des travaux les plus importants ayant mis en évidence l'usage de la programmation par contraintes pour résoudre des jeux. Puis par la suite, proposer un modèle CSP pour la représentation des dynamiques de recherche des stratégies des meilleures réponses, qui servent à former une situation du jeu représente un équilibre de Nash en stratégies pures. Cette modélisation sera en suite illustrée sur des exemples de jeux connus dans la littérature.

Pour présenter ce travail, le présent document est composé de quatre chapitres, où :

- Le premier chapitre est consacré aux rappels théoriques sur quelques éléments essentiels en théorie des jeux.
- Le second chapitre est dédié à la présentation des problèmes de satisfaction de contraintes, en énonçant des définitions de base, le processus de modélisation d'un problème par un CSP et quelques approches classiques de résolution.
- La synthèse bibliographique des principaux travaux ayant appliqués la programmation par contraintes à la théorie des jeux est présentée dans le troisième chapitre.
- Le contenu du dernier chapitre de ce document représente notre contribution. Nous allons présenter ici le modèle CSP proposé pour la résolution d'un jeu sous forme normale et exploiter les outils de la programmation par contraintes dans la résolution et la recherche d'équilibre de Nash du jeu. Nous avons illustré ce modèle proposé sur un jeu connu qui est le dilemme de prisonnier. En seconde lieu, nous avons suit la même procédure pour le cas d'un jeu représenté sous sa forme graphique.

Nous terminons ce travail par une conclusion et des perspectives.

1

Élements de la Théorie des Jeux

1.1 Introduction

La théorie des jeux est une discipline qui a pour but l'étude de situations d'interactions stratégiques, où le sort de chaque participant dépend non seulement de ses propres décision mais aussi des décisions prises par ses adversaires.

La naissance de cette discipline remonte à l'an 1944, suite à l'apparition du livre fondateur " *Theory of Game and Economic Behavior*", du mathématicien *John Von Neumann* et de l'économiste *Oskar Morgenstern*. Par la suite, *John Forbes Nash (1950)* par ses travaux a définit une notion de solution pour les jeux à somme non nulle, ce qui a conforté cette fondation [31]. Depuis, la théorie des jeux a connu un développement mathématique important, faisant d'elle un outil puissant de modélisation et de résolution d'une grande variété de problèmes rencontrés plusieurs domaines d'applications, notamment en économie, en transport et logistique, en télécommunication, en sociologie et en biologie [14].

Dans ce chapitre, nous comptons définir quelques éléments de base de cette théorie, qui vont nous permettre de bien comprendre le contenu de ce mémoire.

1.2 Concepts de base

La théorie des jeux est un outil mathématique qui étudie des situations où les individus dit *"joueurs"* sont conduits à faire des choix parmi un certain nombre d'actions possibles, dans le but d'atteindre un ou plusieurs objectifs bien précis.

En voici quelques définitions élémentaires à connaître en théorie des jeux :

Définition 1.1 (Jeu [33]) *Un jeu est un ensemble de relations entre un ensemble de décideurs en situation d'interaction décisionnelle, où chacun doit faire des choix rationnels sur un ensemble d'actions de sorte à respecter les règles du jeu. Le résultat de ces choix forme une* issue du jeu, *à laquelle est associé une fonction de gain (ou paiement) à chacun des participants.*

Définition 1.2 (Joueurs) *Un jeu comprend un ensemble d'individus rationnels qui peuvent prendre des décisions. Ils interagissent et influent par leur choix sur les résultats des choix de leurs adversaires, cet ensemble d'individus est appelé ensemble des* joueurs. *Un joueur peut être une entreprise, un pays, une région,une cellule,etc.*

Définition 1.3 (Stratégies) *Une stratégie est la spécification complète de comportement d'un joueur dans n'importe quelle situation du jeu. On distingue deux types de stratégie :*

- ***Les stratégies pures :*** *Une stratégie pure est une action ou un plan d'actions choisie avec certitude par le joueur. On notera par :*

  - N *: l'ensemble des joueurs participant au jeu* ($N \geq 2$) *;*
  - $S_i = \{s_{i1}, s_{i2}, ..., s_{in_i}\}$ *: l'ensemble des stratégies pures du joueur* $i \in N$ *;*
  - s_i *: la stratégie pure choisie par le joueur* i *;*
  - n_i *: le nombre de stratégies pures du joueur* i*.*

- ***Les stratégies mixtes*** *Une stratégie mixtes est une distribution de probabilité sur l'ensemble des stratégies pures pour chacun des joueurs. Une stratégie mixte pour un joueur* i *est représentée par un vecteur* Δ_i *défini comme suit :*

$$\Delta_{n_i} = \{\alpha_i = (\alpha_1, ...\alpha_{n_i}) \in \mathbb{R}^{n_i} : 0 \leq \alpha_i \leq 1 \quad \forall j \in \{1, ...n_i\}, \ \sum_{j=1}^{ni} \alpha_i = 1\}$$

  Où n_i *est le nombre de stratégies pures de joueur* i *et* α *représente une stratégie mixte de ce joueur.*

Définition 1.4 (Les fonctions d'utilité [27]) *Elles représentent le gain et le bénéfice négatif (perte) ou positif qui résulte des choix de chaque joueur.*

1.3 Classification des jeux

La classification des jeux se fait suivant différents aspects :

- La somme des gains des joueurs.
- Le déroulement du jeu dans le temps.
- La nature de l'information.
- Le type de relations entre les joueurs.
- Le nombre de coups.

On obtient alors les différentes classes suivantes :

Définition 1.5 (Jeu à somme nulle/non nulle) *Un jeu est dit à somme nulle, si en toute situation du jeu, la somme des gains de tous les joueurs est égale à 0. Dans le cas contraire, il est dit à somme non nulle.*

Définition 1.6 (Jeu à information complète/incomplète) *Un jeu est dit à information complète, si chaque joueur connaît toute la structure du jeu, c'est-à-dire : l'ensemble des joueurs, l'ensemble des stratégies pour chaque joueur, ainsi que leurs fonctions de gains. Si au moins un des joueurs ignore un des éléments structurant le jeu, ce dernier est dit à information incomplète.*

Définition 1.7 (Jeu à information parfaite/imparfaite) *On dit qu'un jeu est à information parfaite, si chaque joueur est parfaitement informé des actions passées des autres joueurs et de leurs conséquences.*
Si au moins un des joueurs ignore certains choix qui ont été effectués avant le sien, alors on parle d'un jeu à information imparfaite.

Définition 1.8 (Jeu statique/dynamique) *On dit q'un jeu est statique, lorsque les joueurs choisissent leurs actions simultanément, c'est-à-dire au même temps.*
On dit qu'un jeu est dynamique, lorsque les joueurs choisissent leurs actions séquentiellement, c'est-à-dire à tour de rôle.

Définition 1.9 (Jeu fini/infini) *Un jeu sous forme normale est dit fini si les ensembles des stratégies ont tous un nombre fini d'éléments.*

S'il existe au moins un joueur $i \in N$ tel que son ensemble de stratégies S_i est infini, le jeu est considéré comme étant infini.

Définition 1.10 (Jeu coopératif/non-coopératif) *Un jeu est dit coopératif lorsque les joueurs peuvent communiquer librement entre eux et passer des accords. Ils forment alors une coalition et recherchent l'intérêt générale suivi d'un partage des gains entre tous les joueurs. Dans le cas contraire, lorsque les joueurs agissent d'une façon opportuniste, le jeu est dit non-coopératif.*

1.4 Représentation d'un jeu

Afin d'étudier tous les jeux, la première question que nous devons nous soucier est comment le représenter. une représentation d'un jeu est une structure de données qui stocke toutes les informations nécessaires pour spécifier un jeu. une représentation d'un jeu est dit être pleinement expressif si elle est capable de représenter n'importe quel jeu arbitraire. Enfin, la taille d'une représentation de ce jeu est la quantité de données pour exprimer une instance de jeu. Dans cette section nous faisons un sondage sur les représentations existantes d'un jeu dans la littérature.[32]

La forme normale

Un jeu sous forme normale est la spécification de l'espace des stratégies et des fonctions de paiement de chaque joueur pour chaque situation du jeu.
Un jeu sous forme normale est définit par le triplet suivant :

$$J = \langle I, \{S_i\}_{i\in I}, \{f_i\}_{i\in I}\rangle \tag{1.1}$$

Avec

1. $I = \{1, ..., N\}$: l'ensemble des joueurs participant au jeu $(N \geq 2)$;
2. $S_i = \{s_{i1}, s_{i2}, ..., s_{in_i}\}$: l'ensemble des stratégies pures du joueur $i \in I$;
3. f_i : la fonction de paiement d'un joueur $i \in I$, associé à chaque profil de stratégies $s \in S$, $S = \prod_{i=1}^{N} S_i$.

De la même manière, on définit la forme normale d'un jeu en stratégies mixtes comme suit :

$$J_m = \langle I, \{\Delta_{m_i}\}_{i\in I}, \{E_i\}_{i\in I}\rangle, \tag{1.2}$$

avec

1. Δ_{m_i} : l'ensemble des stratégies mixtes de joueur $i \in I$;
2. E_i : le gain espéré du joueur $i \in I$.

La forme extensive

Un jeu sous forme extensive est donné par un arbre de jeu contenant un noeud initial, des noeuds de décisions, des noeuds terminaux et des branches reliant chaque noeud à ceux qui le succèdent.
Il est à noter que :

- l'ensemble de joueurs est indexé par $I = \{1, 2, ..., N\}, N \geq 2$,
- pour chaque noeud de décision, on précise le nom du joueur qui doit choisir une stratégie,
- pour chaque joueur, la spécification de l'ensemble des actions permises à chaque noeud où il est susceptible de prendre une décision,
- la spécification des gains de chaque joueur se fait à chaque noeud terminal.

Exemple de jeu sous forme extensive

Le jeu représenté dans la figure est un jeu sous forme extensive, à deux joueurs (joueur1, joueur2), fini, et coopératif. cet arbre est constitué de trois noeuds, chaque noueud d'un niveau correspond à un joueur. Chaque branche partant de noeud correspond à une stratégie de ce joueur.
Dans les feuilles, le premier élément de chaque couple représente l'utilité du joueur 1, tandis que le second élément représente celle de joueur 2.

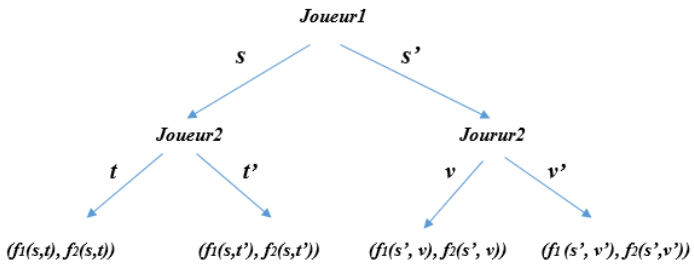

FIGURE 1.1 – Représentation d'un jeu sous forme extensive.

Le joueur 1 commence le jeu et il peut choisir entre deux actions s et $s^{'}$. Si il choisit l'action s et le joueur 2 choisit son action t (resp.$t^{'}$) sachant qu'il connait l'action choisit par le joueur 1, va les mener à des gains

$$(f_1(s,t), f_2(s,t))(\text{resp.}(f_1(s,t^{'}), f_2(s,t'))) \tag{1.3}$$

et si le joueur 1 choisit l'action s' et le joueur 2 choisait son action v (resp.v') sachant qu'il connait l'action choisit par le joueur 1, il aurant les gains

$$(f_1(s',v), f_2(s',v))(resp(f_1(s',v'), f_2(s',v'))) \tag{1.4}$$

et le jeu se termine. Beaucoup de travaux de littérature qui portent un intérêt à l'étude des liens entre la programmation par contraintes et la théorie des jeux évoquent la notion de jeux graphiques. Nous jugeons alors qu'il est judicieux de présenter cette autre manière de représenter des jeux pour pouvoir mieux comprendre le troisième chapitre de ce mémoire. La forme normale et la forme extensive ne sont pas assez concis pour modéliser les jeux statique, on a alors besoin de définir une représentation compacte des fonctions d'utilités. Différentes propositions ont contribué dans ce sens. Une de ces contributions est la définition de la représentation par jeux graphiques. Dans les jeux graphiques, chaque joueur $i \in I$ est représenté par le sommet d'un graphe orienté G, on utilise $N(i) \subseteq P$ pour noter le voisinage du joueur i dans G. Cela signifie que ces sommets j tels que le bord dirigé (i, j) apparaît dans G. $N(i)$ inclut i lui même car l'utilité du joueur i dépend également de sa décision. Soit s une stratégie commune, nous utilisons $\phi_i(s)$ pour désigner la projection de s sur les seuls joueurs de $N(i)$.

Définition 1.11 (Jeu graphique [32]) *Un jeu graphique est une paire (G, M) telle que :*
- *G : est un graphe non orienté dans les sommets $\{1, ..., n\}$;*
- *M : est un ensemble de n matrices de jeu locale.*

Pour chaque stratégie commune s, la matrice de jeu locale $M_i \in M$ spécifie les gains $M_i(\phi_i(s))$ pour le joueur i, qui dépend juste des stratégies choisies par les joueurs dans $N(i)$.

Exemple de jeu sous forme graphique [32]

Adam, Dylan et Julie sont invités à une fête. Adam veut accepter l'invitation. Dylan aussi, mais il ira si et seulement si Adam y va aussi. Pour Julie, elle veut aller avec Dylan mais sans Adam.
L'exemple peut être exprimé par un jeu graphique composé du graphe orienté présenté dans la figure suivante et un ensemble de matrices locales dans le tableau :

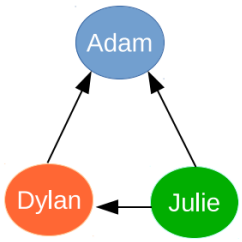

FIGURE 1.2 – Représentation d'un jeu sous forme graphique.

Dans l'exemple, l'utilité d'Adam ne dépend de personne d'autre. Par conséquent, seules ses stratégies apparaissent dans sa matrice locale (le tableau en haut à gauche). Ensuite,

Adam	
Adam	0 (0)
	1 (1)

Dylan	*Dylan*	
	0	1
Adam 0	(0)	(0)
Adam 1	(0)	(1)

Julie (1 (yes))	*Dylan*	
	0	1
Adam 0	(0)	(1)
Adam 1	(0)	(0)

Julie (0 (no))	*Dylan*	
	0	1
Adam 0	(0)	(0)
Adam 1	(0)	(0)

FIGURE 1.3 – Matrices locales de tous les joueurs de l'exemple .

parce que l'utilité de Dylan dépend d'Adam, il y a les stratégies d'Adam et Dylan dans la matrice locale de Dylan (le tableau en haut à droite). Enfin, le dernier la décision de Julie dépend à la fois d'Adam et de Dylan. Par conséquent, les stratégies de tous les acteurs se produisent dans la matrice locale de Julie (les deux tableaux en bas). En effet, au lieu de stocker $3 \times 2^3 = 24$ entrées sous forme normale, il suffit de stocker 2+4+8 = 14 entrées dans les trois matrices locales de jeu graphique. La taille totale de la représentation est $\Theta\Theta(n \times m^{(\xi+1)})$ où ξ est le degré maximal de G.

Les jeux graphiques permettent de réduire la taille des tables utilitaires car ils ne requiert un certain nombre de paramètres exponentiels de la taille d du plus grand voisinage local. Par conséquent, si d $\ll n$, alors la représentation graphique est significativement plus compacte que la forme normale. Cependant, cela signifie également que cette représentation ne convient pas aux jeux qui ne sont pas décomposable, c'est-à-dire lorsqu'il existe une interaction complète entre tous les joueurs dans les jeux en tant que taille retomberait à la taille normale du formulaire.

Les jeux matriciels

Plus adaptée aux jeux à deux joueurs, mais bien que l'on puisse aussi décrire par ce moyen des jeux de plus de deux joueurs, cette representation est directement issue des bases de la théorie des jeux.
On confond souvent jeux sous forme normale et jeux sous forme matricielle,car ils sont trés liés. En génerale, les jeux sous forme normale s'etudient assez simplement lorsque l'on passe au préalable par la représentation matricielle.
Les jeux sous forme matricielle ont pour attrait de permettrede lire rapidement les stratégies dominées. Soit celles qu'il est inutile de jouer parce que d'autres fourniront plus de gain à coup sur. Cette représentation permet de lire rapidement ses intérêts ainsi que ceux des adversaires.[15]

Définition 1.12 (Jeux matriciels [39]) – *Nous avans N joueurs* $I = \{1, 2, ..., N\}$
- *Chaque joueur a un ensemble de stratégie* S_i*. Une stratégie du joueur i est indiqué par* s_i*.*
- *Chaque joueur* $i \in I$ *a une fonction d'utilité* $f_i(s_1, s_2, ...s_N)$*.*

- *Un jeu matriciel J est fini si chaque ensemble de stratégies S_i est fini.*
- *Dans le cas de deux joueurs et jeu fini, nous pouvons représenter le jeu sous forme d'un jeu matriciel : les lignes sont étiquetées avec les stratégies pour le joueur 1, les colonnes avec les stratégies du joueur 2 et la matrice d'entrée M_{ij} est les paires de gains pour la paire de stratégies (s_i, s_j).*

1.5 Concept de solution pour les jeux sous forme normale

En théorie des jeux, un concept de solution peut être défini comme un ensemble de lois représentées par des équations mathématiques, permettant de sélectionner parmi toutes les issues possibles du jeu, un sous-ensemble d'issues satisfaisant certaines propriétés, jugées désirables par les joueurs qui par hypothèse, possèdent certaines facultés de raisonnement ou de comportement (rationalité, prudence, etc).

Notion de stratégies dominantes

Définition 1.13 (Stratégie strictement dominante [46]) *Une stratégie $s'_i \in S_i$ est strictement dominante pour le joueur $i \in I$, si :*

$$f_i(s'_i, s_{-i}) > f_i(s_i, s_{-i}), \qquad \forall s_i \in S_i \;\; \forall s_{-i} \in S_{-i} \tag{1.5}$$

Définition 1.14 (Stratégie strictement dominée [46]) *Une stratégie $s'_i \in S_i$ est strictement dominée pour le joueur $i \in I$, s'il existe une stratégie $s_i \in S_i$ tel que :*

$$f_i(s'_i, s_{-i}) < f_i(s_i, s_{-i}), \qquad \forall s_i \in S_i \quad \forall s_{-i} \in S_{-i} \tag{1.6}$$

Procédure d'élimination des stratégies strictement dominées

Si aucun joueur ne choisit de stratégie strictement dominée, alors les autres joueurs peuvent anticiper ce phénomène, ceci nous amène à définir le processus d'élimination itérée des stratégies strictement dominées.

Le principe de cette procédure consiste à éliminer pour chaque joueur, toutes les stratégies strictement dominées. Ce processus nous permet de réduire au fur et à mesure la taille du jeu (par réduction des ensembles de stratégies des joueurs) tout en faisant apparaître de nouvelles relations de dominance. En éliminant de la même manière les éventuelles stratégies dominées pour le nouveau jeu, on obtient à chaque étape un nouveau jeu plus réduit. On continue ainsi jusqu'à ce que les joueurs n'aient plus de stratégie dominée à éliminer [29]. La procédure itérée peut être décrite comme suit :

- Si un joueur i a une stratégie strictement dominée s_i, il ne devrait pas la jouer.
- Les autres joueurs savent aussi que le joueur i ne devrait pas jouer s_i.
- On peut donc éliminer s_i c'est-à-dire étudier le jeu où l'ensemble de stratégies du joueur i est $S_i \setminus \{s_i\}$.
- Plus formellement : on pose $\Gamma_1 = \Gamma$. A chaque étape $K \geq 1$:
- Si Γ^K n'a pas de stratégies strictement dominées, la procédure s'arrête.
- Sinon, soit $i \in I$ et si ${S^K}_i$ est une stratégie strictement dominée de Γ^K, on pose $\Gamma^{K+1} = (I, S^{K+1}, f)$ avec :
  ${S^{K+1}}_{-i} = {S^K}_{-i}$ et
  ${S^{K+1}}_i = {S^{K+1}}_i \backslash \{s_i\}$.

L'ordre d'élimination des stratégies strictement dominées n'a pas d'influence sur le résultat final.
Si on supprime de manière itérée les stratégies dominées, le résultat peut dépendre de l'ordre des itérations. Ce n'est donc pas une procédure qui va nous permettre de de résoudre le jeu de manière unique. Le jeu sera dit résoluble par élimination itérée de stratégies dominées si pour tout i, S_i^∞ est un singleton, la situation du jeu obtenue à la fin de cette procédure représente un équilibre de Nash en stratégies pures. Dans le cas où un des ensembles S_i n'est pas un singleton, cette procédure va nous permettre d'obtenir un jeu de plus petite taille, donc de complexité réduite.

1.5.1 Équilibre d'un jeu

L'analyse d'un jeu permet de prédire l'équilibre qui émergera si les joueurs sont rationnels. Par équilibre, nous entendons un état ou une situation dans laquelle aucun joueur ne souhaite modifier son comportement compte tenu du comportement des autres joueurs [20].

Définition 1.15 (Équilibre de Nash en stratégies pures [43]) *Une issue*

$$s^\star = (s_i^\star, s_{-i}^\star) \in S = \prod_{i=1}^{N} S_i \tag{1.7}$$

est un équilibre de Nash en stratégies pures pour le jeu (1.1) *si et seulement si :*

$$f_i(s_i{}^\star, s_{-i}{}^\star) \geq f_i(s_i, s_{-i}{}^\star), \quad \forall s_i \in S_i, \forall i \in I \tag{1.8}$$

Définition 1.16 (Équilibre de Nash en stratégies mixte [43]) *Une issue*

$$\alpha^\star \in \Delta = \prod_{i=1}^{N} \Delta_{mi} \tag{1.9}$$

est un équilibre de Nash en stratégies mixtes du jeu (1.2) *si et seulement si :*

$$E_i(\alpha_i{}^\star, \alpha_{-i}{}^\star) \geq E_i(\alpha_i, \alpha_{-i}{}^\star) \quad \forall \alpha_i \in \Delta_{mi}, \quad \forall i \in I \tag{1.10}$$

Théorème 1.1 *Tout jeu fini admet un équilibre de Nash en stratégies mixtes [9].*

Définition 1.17 (Stratégies de meilleures réponses [40]) *Une stratégie est une meilleure réponse d'un joueur, si elle lui offre la plus grande satisfaction face à ce profil. Formellement :*
Une stratégie $s_i \in S_i$ est meilleure réponse aux stratégies pures des autres joueurs S_{-i} dans un jeu sous forme normale si :

$$f_i(s_i, s_{-i}) \geq f_i(s'_i, s_{-i}) \quad \forall s_i \in S_i \tag{1.11}$$

Définition 1.18 (Équilibre de Pareto [40]) *Un optimum de Pareto est un état dans lequel on ne peut pas améliorer le bien-être ou le gain d'un individu sans détériorer celui d'un autre.*

Une issue $s_i \in S_i$ est appelée équilibre de Pareto pour le jeu s'il n'existe pas une autre stratégie $s_i \in S_{-i}$ tel que :

$$f_i(s_i{}^\star, s_{-i}{}^\star) \leq f_i(s_i, s_{-i}) \quad \forall i \in I \tag{1.12}$$

et pour au moins un $j_0 \in I$, on ai :

$$f_{j_0}(s_i{}^\star, s_{-i}{}^\star) < f_{j_0}(s_i, s_{-i}) \tag{1.13}$$

Définition 1.19 (Équilibre admissible) *Un équilibre de Nash est dite admissible s'il n'existe pas un autre équilibre de Nash qui est pareto meilleur.*

Définition 1.20 (Équilibre Parfait) *Un équilibre de Nash est dit parfait en sous-jeux (équilibre parfait) lorsque les stratégies qui le constituent engendre un équilibre de Nash dans tout sous-jeu.*

1.6 Complexité du calcul des équilibres de Nash

Papadimitriou a introduit le concept de *polynomial parity arguments on directed graphs (PPAD)*, une classe de comlexité qui inclut le calcul d'équilibre de nash mixte. Semblable à la classe NP, il existe un ensemble de problèmes $PPAD - complets$ établi qui sont considérés comme étant de nature informatique difficile.
On peut se demander pourquoi la class $PPAD$ est utilisée au lieu de NP pour représenter

l'intransigeance de calculer les équilibres de nash. La réponse est que les problèmes de NP sont classiquement un ensemble de problèmes de décision de la forme "existe-t-il une solution à ce problème ?". Eh bien, le théoreme de Nash garantit l'existance d'un équilibre de Nash mixte, de sorte que le problème du calcul des équilibres ne semble pas bien dans ce cadre du NP.
Nash est PPAD. PPAD est une sous-classe de NP, elle définit par une reduction du problème en question à un graphique dirigé [2].

1.7 Méthodes de calcul des équilibres de Nash

Avant de présenter ces méthodes de calcul, on aura besoin de définir la notion de stratégies prudentes.

Stratégie prudente

Un joueur n'aiment pas le risque, peut choisir de mettre en place une stratégie *prudente*, à savoir une stratégie qui dans le pire des sénarios, lui assure un gain minimum.

Définition 1.21 (Stratégie prudente) *une stratégie ${s^*}_i \in S_i$ est une stratégie prudente ou stratégie maxmin de joueur $i \in I$, si :*

$$ {s^*}_i \in \arg\max_{s_i \in S_i} \min s_{-i} \in S_{-i} \tag{1.14} $$

Autrement dit, ${s^*}_i$ est la stratégie qui assure le meilleur niveau d'utilité garanti possible en stratégies pures pour le joueur i. De la même façon, on peut définir une stratégie mixte prudente.

Programmation Linéaire

La relation entre la théorie des jeux et la programmation linéaire a été évoquée pour la première fois par **Dantzig** [29]. La résolution d'un jeu matriciel revient à résoudre une paire de programmes linéaires duaux, et les solutions optimales d'un des deux programmes correspondent aux stratégies optimales de l'un des deux joueurs. En vertu de théorème minmax, il existe toujours une stratégie mixte de garantie [28] :
$\alpha^* \in \Delta_m$, pour le joueur 1
$\beta^* \in \Delta_n$, pour le joueur 2
M est la matrice des gain du joueur i Et la valeur de jeu est donnée par

$$ \underline{V}_m(M) = \max_{\alpha \in \Delta_m} \min_{\beta \in \Delta_n} \alpha^T M \beta = {\alpha^*}^T M \beta = \min_{\beta \in \Delta_n} \max_{\alpha \in \Delta_m} \alpha^T M \beta = \overline{V}_m(M) $$

Et en vertu résultat de la paire stratégies (α^*, β^*) constitue un équilibre point selle pour

le jeu (1.2). nous avons :

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}\alpha_i{\beta_j}^* \leq \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}{\alpha_i}^*{\beta_j}^* \leq \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}{\alpha_i}^*\beta_j \qquad \forall\alpha\in\Delta_m, \forall\beta\in\Delta_n$$

$$\begin{cases} \min\limits_{\beta\in\Delta_n}\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} f_{ij}\alpha_i\beta_j \to \max \\ \sum\limits_{i=1}^{m}\alpha_i = 1, \\ \alpha_i \geq 0, \forall i = \overline{1,m} \end{cases} \tag{1.15}$$

$$\begin{cases} \max\limits_{\alpha\in\Delta_m}\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} f_{ij}\alpha_i\beta_j \to \min \\ \sum\limits_{i=1}^{m}\beta_j = 1, \\ \beta_j \geq 0, \forall j = \overline{1,n} \end{cases} \tag{1.16}$$

et on a :

$$\min_{\beta\in\Delta_n}\alpha^T M\beta = \min_{j\in 1,\ldots,n}\alpha^T M{\cdot}j = \min_{j=1,\ldots,n}\sum_{i=1}^{m} f_{ij}\alpha_i \ \forall\alpha\in\Delta_m \max_{\alpha\in\Delta_m}\alpha^T M\beta = \max_{i\in 1,\ldots,m} M^T i{\cdot}\beta = \max_{i=1,\ldots,m}\sum_{j=1}^{n} f_{ij}\beta_j \ \forall\beta\in\Delta_n$$

Les deux problèmes (1.14) et (1.15) se transforment respectivement aux deux problèmes suivant :

$$\begin{cases} \min\limits_{j\in 1,\ldots,n}\sum\limits_{i=1}^{m} f_{ij}\alpha_i\beta_j \to \max \\ \sum\limits_{i=1}^{m}\alpha_i = 1, \\ \alpha_i \geq 0, \forall i = \overline{1,m} \end{cases} \tag{1.17}$$

et

$$\begin{cases} \max\limits_{i\in 1,\ldots,m}\sum\limits_{j=1}^{n} f_{ij}\alpha_i\beta_j \to \min \\ \sum\limits_{j=1}^{n}\beta_j = 1, \\ \beta_i \geq 0, \forall j = \overline{1,n} \end{cases} \tag{1.18}$$

Les deux programmes (1.16) (1.17) sont équivalents à les deux programmes suivant :

$$\begin{cases} \xi \to \max \sum_{i=1}^{m} f_{ij}\alpha_i \geq \xi, \forall j = \overline{1,n} \\ \sum_{i=1}^{m} \alpha_i = 1, \\ \alpha_i \geq 0, \forall i = \overline{1,m} \end{cases} \tag{1.19}$$

$$\begin{cases} \eta \to \min \sum_{j=1}^{n} f_{ij}\beta_j \leq \eta, \forall i = \overline{1,m} \\ \sum_{j=1}^{n} \beta_i = 1, \\ \beta_j \geq 0, \forall j = \overline{1,n} \end{cases} \tag{1.20}$$

α^* (resp β^*) est une solution optimale du problème(1.14) (resp(1.15))
Les problème (1.18) (1.19) sont une paire de problème de programmation linéaire duaux. Comme tout jeu matriciel admet un point selle et une valeur de jeu en stratégies mixtes, alors la résolution des deux problèmes duaux (1.18) (1.19) permettent d'obtenir, d'aprés le théorème fort de dualité,

$$\xi^* = \eta^* = V_m(M), \text{ où } \xi^* \text{ et } \eta^*$$

sont les valeurs optimales des problèmes respectivement (1.18) et (1.19).
de plus,

$$\xi^* = \underline{V}_m(M) = \max_{\alpha\in\Delta_m}\min_{\beta\in\Delta_n} \alpha^T M\beta = \min_{\beta\in\Delta_n} \alpha^{*T} M\beta = \alpha^{*T} M\beta^*$$

$$\eta^* = \overline{V}_m(M) = \min_{\beta\in\Delta_n}\max_{\alpha\in\Delta_m} \alpha^T M\beta = \max_{\alpha\in\Delta_m} \alpha^T M\beta^* = \alpha^{*T} M\beta^*$$

L'algoritheme de Lemke-Howson

Proposition 1.1 *[1]*

Une stratégie mixte est une meilleur réponse par rapport à un ensemble de stratégies S si et seulement si toutes les stratégies de son support sont aussi des meilleurs réponses par rapport à S.

Actuellement, l'algorithme de Lemke-Howson est l'algorithme combinatoire le plus efficace pour calculer les équilibres de Nash mixtes. L'idée de l'algorithme est de parcourir l'espace des profils en se basant sur les propriétées des meilleures réponses (notamment sur la propriétés des supports des équilibres de Nash). A chaque itération, le support de l'état

courant est modifié légèrement afin d'accéder vers un état ayant un support qui "respecte le plus" la propriété 1. Remarquons que cet algorithme ressemble beaucoup à la méthode du simplexe de la programmation linéaire puisque la modification du support est basée sur une succession d'applications d'une méthode du pivot pour résoudre des systémes linéaires.

Du point de vue de la théorie des graphes, l'algorithme de Lemke-Howson parcourt un graphe dont chaque sommet correspond à un profil du jeu et dont chaque sommet a un degré sortant et un degré rentrant qui vaut 0 ou 1. En effet, cela correspond au fait que la modification d'un support d'un profil vers un autre est déterministe. De plus, les équilibres de Nash correspondent à des puits (aucune meilleure modification du support est possible).

Remarquons que le nombre de sommets du graphe peut être exponentiel. Un point crucial de cet algorithme est que connaître le voisinage d'un sommet se réalise en un temps polynomial. L'algorithme construit ainsi le graphe à la volée.
Par contre, il peut explorer un nombre exponentiel de sommets même pour des jeux particuliers comme les jeux symétriques à 2 joueurs.[1]

1.8 Conclusion

Nous avons décrit dans ce chapitre quelques notions de base sur la théorie des jeux ainsi que quelques concepts de solutions. Un intérêt particulier a été porté au concept d'équilibre de Nash, vu que d'une part, c'est la notion de solution la plus étudié en théorie des jeux, d'autre part, nous envisageons de voir comment cet équilibre peut être calculé en utilisant la programmation par contraintes. La présentation de cette discipline va faire l'objet du chapitre suivant.

2

La Programmation par Contraintes (PPC)

2.1 Introduction

La programmation par contraintes est une approche de résolution des problèmes combinatoires complexes issus de la programmation logique et de l'intelligence artificielle. Elle opère en modélisant le problème sous forme d'un ensemble de relations logiques qui définissent l'ensemble des valeurs autorisées pour les variables de décision du problème [21].

Le présent chapitre a pour but de présenter les notions de base de la programmation par contraintes, et de donner un aperçu sur les méthodes utilisées pour la résolution d'un problème de satisfaction de contraintes (CSP).

2.2 Les problèmes de satisfaction de contraintes (CSP)

2.2.1 Définitions élémentaires

Les problèmes de satisfaction de contraintes ont été définis initialement par *MONTANARI* en 1974. Ce formalisme vise à représenter sous forme de contraintes, les propriétés et les relations qui existent entre les objets manipulés. Ces contraintes peuvent être décrites par de multiples façons (une équation, une inégalité, un prédicat, une fonction booléenne, une énumération des combinaisons de valeurs autorisées, etc). Elles traduisent l'autorisation ou l'interdiction d'une combinaison de valeurs.

D'une manière générale, une instance d'un problème de satisfaction de contraintes est défini par l'ensemble des variables du problème, le domaine indiquant les valeurs possibles

pour chaque variable, et l'ensemble des contraintes du CSP. Formellement :

Définition 2.1 (Le problème CSP) *[36] Un problème de satisfaction de contraintes* $(\mathcal{P})$ *est défini par un triplet* (X, D, C)*, où :*

- $X = \{X_1, X_2, ..., X_n\}$ *est un ensemble de n variables de décision.*
- $D = \{D_1, D_2, ..., D_n\}$ *est un ensemble de domaines finis, où* D_i *est le domaine associé à la variable* X_i*. Il représente l'ensemble de ses valeurs possibles.*
- $C = \{C_1, C_2, ..., C_m\}$ *est un ensemble de m contraintes, où une contrainte* C_i *est définie sur un sous ensemble de variables.*

On distingue trois grandes catégories de problèmes de satisfaction de contraintes :

1. **Les CSP discrets :** Dans cette catégorie de problèmes, les variables de décision sont discrètes.
2. **Les CSP continus :** Il s'agit ici du cas où les variables sont à valeurs continues.
3. **Les CSP mixtes :** Ils concernent à la fois des variables continues et discrètes.

Définition 2.2 (Les contraintes [37]) *Une contrainte est une relation logique établie entre différentes variables, chacune prenant sa valeur dans son domaine. Une contrainte sur un ensemble de variables restreint les valeurs que peuvent prendre simultanément ses variables.*

Définition 2.3 (La portée d'une contrainte [36]) *La notion de portée d'une contrainte fait référence à l'ensemble des variables concernées par la contrainte.*

Définition 2.4 (L'arité d'une contrainte [36]) *L'arité d'une contrainte représente le nombre de variables de décision concernées par cette contrainte. Selon l'arité de la contrainte, nous distinguons :*

- *Les contraintes unaires : c'est des contraintes dont l'arité est égale à 1.*
- *Les contraintes binaires : c'est des contraintes dont l'arité est égale à 2.*
- *Les contraintes n-aires : c'est lorsque l'arité de la contrainte est supérieure ou égale à trois.*

Définition 2.5 (Instanciation [36]) *Une instanciation I d'un ensemble* $X = \{X_1, ..., X_k\}$ *de k variables est un ensemble* $\{(x_1, v_1), ..., (x_k, v_k)\}$ *tel que* $v_1 \in D_1, ..., v_k \in D_k$*.*

∘ *Une instanciation est dite totale, si elle attribue une valeur à toutes les variables du problème, sinon elle est dite partielle.*
∘ *Si l'instanciation ne viole aucune contrainte (ç'est-à-dire qu'elle satisfait toutes les contraintes portant sur les variables instanciées), elle est dite consistante.*

Définition 2.6 (Solution d'un CSP [8]) *Une solution d'un CSP est une instanciation consistante des variables de X sur D.*

2.2.2 Représentation graphique d'un CSP

Le graphe de contraintes d'un CSP binaire est un graphe simple, où les sommets représentent les variables et les arêtes représentent les contraintes. L'existence d'une arête entre deux sommets est conditionnée par le fait que les variables correspondantes à ces sommets soient liées ou pas par une contrainte dans le problème CSP en question.

Le concept de graphe de contraintes peut être défini pour des CSPs quelconques (non binaire), les contraintes ne sont alors pas représentées par des arêtes mais plutôt par des hyper-arêtes qui relient des sous ensembles de variables impliquées dans une même contraintes. Dans ce cas là, on parle d'hypergraphe de contraintes [14].

***Exemple* 2.1** *Soit P un CSP discret définit par le triplet :*

1. $X = \{X_1, X_2, X_3, X_4, X_5\}$;
2. $D = \{D_1, D_2, D_3, D_4, D_5\}$, $D_1 = D_2 = \{a, b, c\}$ *et* $D_3 = D_4 = D_5 = \{a, b\}$;
3. $C = \{C_{13}, C_{23}, C_{34}, C_{35}, C_{45}\}$, $C_{13} = C_{23} = \{(b, a), (c, a), (c, b)\}$, $C_{34} = C_{35} = \{(a, b)\}$ *et* $C_{45} = \{(a, b)\}$.

La notation C_{ij} signifie que la contrainte est imposé sur les valeurs des deux variables X_i et X_j.

▷ *Le CSP P peut être représenté sous forme d'un graphe $G = (S, A)$, tel que :*

1. $S = \{s_1, s_2, s_3, s_4, s_5\}$ *est l'ensemble des sommets du graphe qui représentent les variables du CSP ;*
2. $A = \{(s_1, s_3), (s_2, s_3), (s_3, s_4), (s_3, s_5), (s_4, s_5)\}$ *est l'ensemble des arrêtes du graphe, il correspond à l'ensemble des contraintes du CSP P.*

La figure 2.1 est une représentation graphique de G.

▷ *Une solution du problème est une instanciation qui satisfait simultanément toutes les contraintes.*

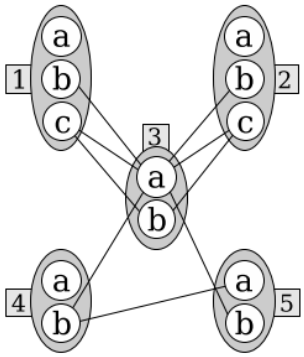

FIGURE 2.1 – Graphe de contraintes du CSP P.

2.2.3 Le problème de satisfiabilité (SAT) :

Le problème SAT est un problème de décision qui consiste à déterminer si une formule booléenne mise sous forme normale conjonctive (CNF), (c'est-à-dire sous la forme d'une conjonction de disjonction de littéraux) admet ou non une solution.

Définition 2.7 (Le problème SAT) *Un problème de satisfiabilité P, dit problème SAT, est défini par un couple(X, F) où :*

- *$X = (X_1, ..., X_n)$ est une séquence de n variable booléennes.*
- *F est une formule définie comme une conjonction de m clauses c_j, tel que : $F = c_1 \wedge ... \wedge c_m$. Chaque clause est définie comme une disjonction de variables X_i ou de leur négation $\overline{X}_i$. Ainsi, $c_j = y_1 \vee ... \vee y_k$, où $y_i \in \{X_1, ..., X_n, \overline{X}_1, ..., \overline{X}_n\}$.*

▶ Une formule F est satisfiable, s'il existe n valeurs pour les variables $\{X_1, ..., X_n\}$ qui rendent la formule F vraie, donc qui rendent toutes les clauses c_i vraie. La recherche de ces valeurs particulières correspond à la résolution du problème SAT.

▶ Le problème CSP peut être vu comme une généralisation du problème SAT. Il a été démontré que toute instance de problème CSP peut être transformé en une instance équivalente de problème SAT. L'idée de base de cette transformation est d'associer une variable booléenne à chaque paire variable-valeur de l'instance CSP. Il suffit par la suite d'associer une clause SAT à chaque combinaison de valeurs interdites par une contraintes CSP [38].

2.2.4 La complexité des problèmes de satisfaction de contraintes

Le problème *SAT* a été le premier problème dont la NP-complétude a été démontrée par *Cook* et *Levin* [7]. Depuis, il représente le problème de référence en théorie de la com-

plexité, pour l'étude de la NP-difficulté des problèmes d'optimisation (académiques et réels) [45, 17]. De la possibilité de conversion de tout problème CSP en problème *SAT* (et inversement) découle naturellement leur appartenance à la même classe de complexité. Par conséquent, le problème CSP est lui aussi un problème NP-complet [6].

Des résultats supplémentaires stipulent que c'est l'étude de la satisfiabilité d'un problème CSP qui est NP-complet. Par contre, la recherche d'une solution admissible ou optimale est un problème NP-difficile [22].

2.2.5 Modélisation des problèmes par des CSPs

Pour modéliser un problème soue forme d'un CSP, il faut d'abord :

1. Identifier les variables ou les inconues.
2. Identifier les domaines de valeurs de ces variables.
3. Identifier les contraintes.

***Exemple* 2.2 (Problème de coloriage de graphe [4])** *Un des problèmes les plus fréquemment utilisés pour expliquer certains concepts et algorithmes de résolutions des CSP est le problème de coloriage de graphe. Ce problème joue aussi un rôle important en recherche opérationnelle, car de nombreux problèmes d'emploi du temps ou d'ordonnancement peuvent être exprimés comme des problèmes de coloriage de graphe. Un problème de coloriage de graphe est décrit comme suit :*

" Etant donné un graphe et un certain nombre de couleurs, il s'agit d'affecter une couleur à chaque noeud tel que deux noeuds adjacents n'aient pas la même couleur."

Une instance de ce problème est le problème de coloriage de cartes. Dans ce cas précis, le problème consiste à affecter une couleur à chaque zone d'une carte géographique en utilisant un nombre limité de couleurs et en respectant la contrainte que deux régions voisines n'aient pas la même couleur. En effet, considérons la carte illustré dans la figure 2.2.

Cette carte peut être représentée par un graphe où chaque noeud représente une zone (région) de la carte et chaque arc indique que les deux noeuds qu'il connecte représentent deux zones voisines, tel que c'est représenté dans la figure 2.3.

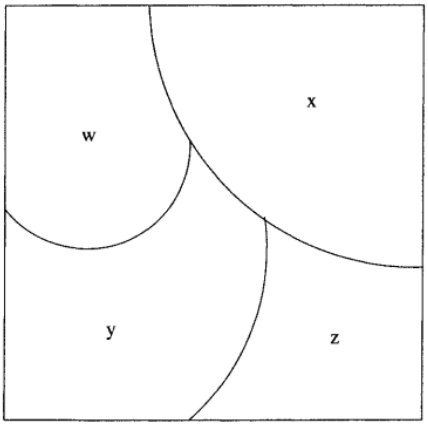

FIGURE 2.2 – Problème de coloriage de carte géographique

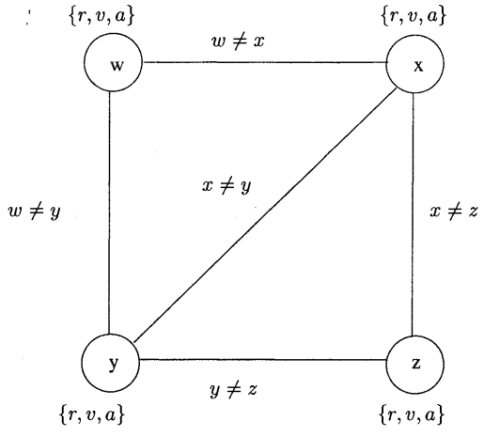

FIGURE 2.3 – Représentation du problème de coloriage de carte sous forme d'un graphe

En supposant que chaque zone géographique peut être coloriée par l'une des couleurs : rouge(r), vert(v), azur(a), la formulation du problème de coloration de carte ci-dessus en tant que problème CSP sera alors :

- $X = \{w, x, y, z\}$;
- $D_w = D_x = D_y = D_z = \{r, v, a\}$;
- $C = \{w \neq x, w \neq y, x \neq z, y \neq z, x \neq y\}$.

2.3 Les méthodes de résolution des problèmes CSP

Une méthode de résolution de CSP doit être en mesure d'identifier une affectation de valeurs aux variables de telle sorte que toutes les contraintes imposées soient satisfaites (ie, une affectation totale et consistance). La démarche générale consiste à explorer l'espace de recherche défini par le produit cartésien des domaines des variables de décision, jusqu'à ce qu'une affectation consistante soit trouvée, ou bien jusqu'à ce qu'à ce que l'inexistence d'une telle affectation soit prouvée (cas d'un CSP inconsistant).

Les notions définies ci-dessous sont très utilisées pour évaluer la qualité d'une instanciation. En effet, de nombreux algorithmes de résolution des CSP se basent sur ces concepts.

Définition 2.8 (L'arc consistance [40]) *Un CSP* (P) *est dit* arc consistant*, si pour tout couple de variables* $(X_i, X_j) \in X^2$*, et pour toute valeur* $v_i \in D_i$ *, il existe une valeur* $v_j \in D_j$ *telle que l'affectation partielle* $\{(X_i = v_i), (X_j = v_j)\}$ *satisfait toutes les contraintes binaires de* C*, liants les deux variables* X_i *et* X_j*.*

Définition 2.9 (La consistance de noeud [40]) *Un CSP* (P) *est dit* noeud consistant*, si pour toute variable* $X_i \in X$*, et pour toute valeur* $v_i \in D_i$ *, l'affectation partielle* $\{(X_i = v_i)\}$ *satisfait toutes les contraintes unaires de* C*, liées à la variable* X_i*.*

Définition 2.10 (k-consistance [40]) *Une affectation consistante* I*, d'un ensemble* $V \subseteq X$ *de* $(k-1)$ *variables, est dite* k-consistante*, si et seulement si pour toute variable* $X_k \in X \backslash V$*, il existe une valeur* $v_k \in D_k$ *telle que l'extension* $(I, X_k = v_k)$ *est consistante. Si toutes les affectations consistantes de* $(k-1)$ *variables sont* k*-consistantes, alors le CSP(P) est dit* k-consistant*.*

***Remarque* 2.1** *Pour un problème de satisfaction de contraintes, on peut ne chercher qu'une solution, ou bien toutes les solutions, alors que pour les problèmes d'optimisation, il s'agit de trouver la meilleure solution en fonction d'un (ou plusieurs) critère(s).*

Dans ce qui suit, nous allons aborder les techniques classiques pour la résolution des problèmes de satisfaction de contraintes.

2.3.1 Générer et tester

Une méthode simple pour chercher une solution d'un CSP est de générer toutes les configurations possibles, ç'est-à-dire toutes les combinaisons possibles de valeurs des variables et de tester si elles vérifient les contraintes du problème en question, cette approche

est connue sous le nom de *Générer et tester*. Le nombre de possibilité testés est alors égale au cardinale du produit cartisien des domaines des variables [41].

L'algorithme Générer et Tester

Le principe de "Générer et tester" consiste à énumérer toutes les affectations possibles jusqu'à en trouver une qui satisfasse toutes les contraintes. Ce principe est repris dans la fonction récursive "GenTest$(A,(X,D,C))$" décrite ci-dessous. Dans cette fonction, A contient une affectation partielle et (X,D,C) décrit le CSP à résoudre (au premier appel de cette fonction, l'affectation partielle est vide). La fonction retourne *"vrai"* si on peut étendre l'affectation partielle A en une affectation totale consistante (une solution), sinon elle retourne *"faux"*.

Cette stratégie est trés coûteuse, d'où l'intérêt de faire appel à des schémas élaborés plus sophistiqués pour résoudre d'une manière plus efficace le problème de satisfaction de contraintes.

```
Algo  GenTest
Entrée:X,D,C: un CSP de taille n;
       A une affectation partielle de taille K<=n
Sortie:B booleen                    % vrai si A peut être etendue en une solution, faux sinon
```

```
B:= false;
if K=n then
    if tester(A,K,C) then B:= true
       else                                        affectation partielle
           K=K+1                                       extension de l'affectation A
            While non B et "nouveau" v dans Dk do
              choisir un "nouveau" v dans Dk,
              A:= Ajout (A,v,K)            par une affectation pour x de (k+1)
              B:= GenTest(X,D,C,A,K )
           end While
     end else
```

2.3.2 Les méthodes par retours-arrière (backtrack)

Afin d'améliorer le comportement naif de l'algorithme "Générer et tester", l'algorithme de recherche généralement utilisé pour résoudre des CSP est celui du **backtrack**. Le principe de cet algorithme consiste à vérifier successivement les ensembles des affectations de variables jusqu'à trouver une solution. Cet algorithme est fondé sur les points de retour. Chaque fois qu'un choix est effectué, un point de retour est mentionné dans l'algorithme. L'ordre des variable à instancier est choisi d'une façon aléatoire ainsi que l'ordre des valeurs à leur attribuer. En d'autre termes, les variables du CSP sont instanciées jusqu'à ce que :

- au moins une contrainte soit violée. L'algorithme revient sur la dernière instanciation, et teste une autre valeur sur la variable la plus récente.
- si toutes les valeurs de cette variable ont été testées sans succès, l'algorithme fait un retour arrière sur la variable qui précède la variable la plus récente [31].
- ce processus s'arrête une fois qu'une solution est trouvée, ou bien le retour arrière atteint la racine de l'arbre de recherche sans qu'une solution ne soit trouvée.

```
Algo  Backtrack-Rec
Entrée:X,D,C: CSP de taille n;
        A une affectation partielle de taille K<= n
Sortie: B booleen
```

```
B false;
if tester(A,K,C) then
    if K=n then B:= true else
          K:= K+1
             While non B et "nouveau" v dans Dk do
               choisir un "nouveau" v dans Dk,
               A:= Ajout (A,v,K)
              B:= Backtrack-Rec(X,D,C,A,K )
            endWhile
       endelse;
endthen
```

2.3.3 Les techniques de filtrage et de propagation de contraintes

Devant la difficulté du problème CSP, il peut se révéler intéressant de réduire l'espace de recherche à explorer en simplifiant les instances avant d'entamer leur résolution. La simplification par filtrage consiste à supprimer des combinaisons de valeurs qui ne peuvent pas participer à une solution. Les combinaisons de valeurs à supprimer dépendent de la forme de consistance locale employée.

Les algorithmes de propagation de contraintes, appelés aussi algorithmes d'arc consistance (AC) sont parmi les approches les plus utilisés. Ils consistent pour chaque contrainte c du problème, à calculer la consistance du CSP réduit à cette seule contrainte, en d'autres termes à retirer des domaines des variables concernées par c chaque valeur pour laquelle, au vu des domaines des autres variables, il n'existe pas de n-uplet compatible avec les domaines des autres variables et permettant d'affecter cette valeur.

La propagation de contraintes est parfois utilisée comme phase de pré-traitement où l'on supprimera certaines valeurs inconsistantes des domaines, avant de lancer un algorithme de retour arrière [32].

2.3.4 Le Forward-Checking

Une technique prospective simple pour anticiper les effets d'une instanciation est nommée le Forward checking. On vérifie que les variables non instanciées peuvent chacune prendre une valeur consistante lorsque l'affectation partielle courante est étendue par l'instanciation d'une nouvelle variable. Les techniques de Forward checking considèrent une seule variables non instanciée.

Le forward checking applique une forme partielle d'arc-cohérence sur le réseau de contrainte après chaque décision [42].

2.4 Les heuristiques de choix lors de recherche d'une solution

Une heuristique de recherche est une fonction/procédure qui a pour but de guider la recherche vers les espaces les plus prometteurs. Dans la littérature, on note la présence de deux grands types. Le premier est fondé sur le choix de variables, la seconde est fondée sur le choix de valeurs lors d'une instanciation d'une variable.

Heuristiques de choix de variables

Nous distinguons :

Les heuristiques statiques :

Dans ce type, l'ordre d'assignation des variables est prédéfini avant le processus de recherche. Deux heuristiques classiques fondées sur les degrés de variables sont :

1. **Max degree :** Dans cette heuristique les variables sont ordonnées selon l'ordre décroissant de leurs degré d'apparition dans les contraintes.
2. **Min degree :** Cette heuristique est construite en inversant le principe de max degree, elle consiste à ordonner les variables selon l'ordre croissant de leurs degré d'apparition dans les contraintes.

Les heuristiques dynamiques :

Dans ce type d'heuristique, le choix de variable s'effectue en se basant sur l'état courant du réseau. Parmi ces heuristiques, on cite :

1. **Domaine et degré (dom+deg)[12] :** Choisir la variable qui a le plus petit domaine courant. Dans le cas d'égalité, choisir la variable avec le plus grand degré.
2. **Domaine sur degré (dom/deg)[5] :** Choisir la variable qui donne le rapport minimale entre la taille du domaine courant et le degré de la variable.

Heuristiques de choix de valeurs

Ce type d'heuristique consiste à guider la recherche lors d'une assignation d'une variable pour réduire l'espace de recherche. Dans la littérature, il existe plusieurs heuristiques sur le choix de valeurs. Nous pouvons citer par exemple, l'heuristique de minimisation de conflits. [25] Pour expliquer brièvement cette heuristique, supposons qu'une instanciation représente un conflit (viole au moins une contrainte). Dans ce cas, l'heuristique choisi arbitrairement une variable parmi celles qui sont en conflit. Puis, parmi toute les valeurs possibles pour cette variable, l'heuristique affecte à cette dernière la valeur qui minimise le nombre de conflit.

2.5 Conclusion

Dans ce chapitre nous avons présenté le formalisme CSP ainsi que les différentes notions de base qui lui sont liées. Nous avons également abordé les méthodes de résolution des CSP.

3

Applications de la PPC dans le calcul des équilibres de Nash

Introduction

L'équilibre de Nash est l'un des meilleur solution pour un jeu, mais la compléxité de la representation des gains des joueurs est l'un des problème aussi qui rend la recherche de cet équilibre défficile. Parce que la bimatrice des gains(la forme normale) croitre exponnontiellement avec le nombre des joueurs.
Ce problème a été abordé par plusieurs types de representation compactes. Certains sont basés sur des hypothèses sur les intéractions entre les joueurs comme les jeux graphiques, tandis que d'autres sont basés sur le langage, comme jeux booléens ou jeux de contraintes.[3]

Ce chapitre présente une synthèse des traveaux les plus pertinents qui se sont intéressés à l'étude des liens entre la théorie des jeux et la programmation par contraintes. Nous allons aborder la question du calcul des équilibres en passant par la résolution de problèmes CSP.

L'approche CSP pour la recherche des équilibres d'un jeu

Les problèmes de satisfaction de contraintes (CSPs) représentent un outil efficace pour la modélisation d'une grande classe de problème rencontré dans plusieurs domaines. plus précisément les problèmes de la théorie des jeux. grace aux équivalences qui existent entre les notions de ces deux théorie, et qui permettent d'exploiter les outils de calcul connus en

programmation par contraintes pour résoudre des problème de la théorie des jeux.

3.1 Les jeux graphique et les CSPs

Un jeu graphique est représenté par un graphe où les noeuds correspondent aux joueurs, et les arêtes entre deux noeuds (i.e. joueurs) représentent l'influence mutuelle que les choix d'un joueurs exercent sur les gains d'un autre joueur.

Les jeux graphiques permettent de représenter les jeux sous forme normale plus compacte, en se basant sur la localité des joueurs, ce qui permette de réduire l'éspace exponentiel requit lors du calcul d'équilibre de Nash qui est un problème difficile même dans le cas des jeux graphiques. Or, l'approche CSP est connue pour être un outil trés puissant pour la résolution des problème difficiles, ce qui explique l'utilisation de cette approche pour la recherche d'equilibre de Nash pure des jeux graphique.

Transformation d'un jeu en un CSP

Considérons le jeu graphique $G_j = \langle G, S, M \rangle$, et soit le problème de satisfaction de contraintes $P = \langle X, D, C \rangle$ dont ses élement sont définit comme suit :

1. $X = \{X_1, \ldots, X_n\}$ est l'ensemble des variables de décision. Chaque variable X_i représente la stratégie pure choisie par un joueur $i \in I$ face à la combinaison de stratégies $S_{N(i)}$ jouée par ses voisins, $X_i = s_i$;
2. $D = \{D_1, \ldots, D_n\}$ est l'ensemble des domaines des variables de décision. Chaque domaine D_i coincide avec l'ensemble des stratégies du joueur associé à la variable X_i, i.e $D_i = S_i,\ i \in I$;
3. $C = \{C_1, \ldots, C_n\}$ est l'ensemble des contraintes du problèmes qui est composé de :

   - Contraintes unaires : Ces contraintes sont définiés sur le profile de stratégie dans le domaine de chaque joueur. c'est à dire que la stratégie s_i de joueur i est une meilleur réponse à la combinaison des statégies $S_{N(i)}$ de ses voisins.
   - Contraintes binaires : Ces contraintes sont définiés sur des pairs de profiles dans le domaine des joueurs voisins. C'est à dire, si i et j sont des joueurs voisins, le joueur i va choisir parmi les stratégies vérifiant les contraintes unaires, celle qui soit meilleure réponse au joueur j, et le joueur j doit agir de la même façon vis-à-vis du joueur i et du reste de ses voisins.
4. Progapation de contraintes : assignez une valeur à chaque variable à tour de rôle, tout en vérifiant toutes les contraintes. De toute évidance, si une affectation à chaque variable peut satisfaire toutes les contraintes, c'est un équilibre de Nash pure pour le jeu correspondant, sinon le jeu n'a pas un équilibre de Nash.

L'approche de la formulation d'un jeu graphique sous forme de CSP a également été examiné par Vickrey et Koller [44]. En cela travail, chaque joueur du jeu est une variable dans le CSP et le domaine de chaque variable est l'espace de stratégie du correspondant au joueur. Les contraintes sur les variables garantissent que aucun joueur ne peut bénéficier d'un écart par rapport à un profil stratégique de son voisinage.

Dans le cadre des jeux graphique toujours, M.jiang et al ont traité un cas spécifique dans [24] . ils ont furmulé le cas de jeu de parité comme CSP et ils ont proposé un algorithme de résolution qui a pu réduire l'espace de recherche des meilleurs réponses. ils ont consillé de prendre en future, cette approche comme un outil pour régler d'autres formalismes bien acceptés et les jeux, en particulier, la logique temporelle modale et le jeu de parité moyenne-rémunération.

Algorithme de résolution

Jiang [16] a proposé un algorithme pour la résolution du $CSP(G_J)$. cet algorithme prend en entrée le jeu sous sa forme graphique, et fournit pour résultat un équilibre de Nash en stratégies pures du jeu, dans le cas où il existe.

◦ Globalement, les étapes de cet algorithme se résument à :

Étape 1 : sélectionner arbitrairement une variable X_i parmi l'ensemble $\{X_1, \ldots, X_n\}$, pour être racine de l'arborescence du graphe, puis choisir un séquencement aux variables restantes et former ainsi une arborescence.

Étape 2 : appliquer les contraintes unaires sur chacune des variables X_i, afin d'éliminer toutes les stratégies qu'un joueur rationnel ne va sûrement pas choisir, donc il va garder uniquement les stratégies de meilleures réponses. Cette étape permet de réduire l'espace de recherche.

Étape 3 : considérer chacune des variables X_i à partir des feuilles de l'arborescence à sa racine et mettre ses voisins dans une file d'attente.

- Tant que ces files d'attente ne sont pas vides, sélectionner un voisin X_j pour chaque variable X_i, puis appliquer les contraintes binaires au couple (X_i, X_j). Ceci permet de réduire les domaines D_i et D_j. Si à une certaine étape, un des domaines se vide, on conclut que le jeu (G_J) n'admet pas d'équilibre de Nash en stratégies pures.
- Affecter une valeur à chaque variable X_i en allant de la racine aux feuilles de façon à conserver la consistance avec la variable précédemment instanciée. Cette propagation des contraintes permet d'accélérer le processus de recherche.

La plupart des travaux permettant de représenter des jeux de façon compacte utilisent la notion de dépendance entre les joueurs et/ou les actions. Certains d'entre eux, dont [10, 23, 30], partagent le mode de représentation des utilités des joueurs suivant : l'utilité du joueur i est décrite par une table associant une valeur numérique à chaque combinaison

de valeurs de chacun des ensembles de variables utiles à i.

La représentation des jeux avec de telles tables d'utilité est appelée forme normale graphique (GNF) dans [13]. Les dépendances entre joueurs et variables entraînent naturellement une relation de dépendance entre joueurs, identique à notre notion de dépendance : i dépend de j si la table d'utilité de i fait référence à une variable contrôlée par le joueur j. Dans [10, 13] les noeuds d'un graphe représentent les agents du jeu, et il y a un arc de i à j si i dépend de j. Cette représentation est plus compacte que la forme normale du jeu si le graphe n'est pas une clique. La notion de dépendance entre joueurs et variables dans les jeux graphiques est utilisée pour exactement la même raison que notre graphe de dépendance, à savoir fractionner un jeu en un ensemble de plus petits jeux en interaction, et pouvant être résolus plus ou moins indépendamment.

3.2 Les CP-nets et équilibre de Nash

Les CP-nets, sont un autre langage de représentation compacte, et sont utilisés dans [18], pour représenter des jeux : les CP-nets sont vus comme des jeux en forme normale et vice versa. Chaque joueur i est associé à une variable X_i du CP-net, dont le domaine est l'ensemble des actions possibles du joueur. Les préférences sur les actions d'un joueur étant données les stratégies des autres joueurs sont exprimées dans une table de préférences conditionnelles. Afin de pouvoir exprimer les préférences d'un joueur dans des tables de préférences conditionnelles, les relations de préférences dans un jeu stratégiques sont paramétrées par les stratégies des autres joueurs.

Définition 3.1 *[18]. Un jeu stratégique avec préférences paramétrées est un tuple* $G = (N, S_1, ..., S_n, \succ (s_{-1}), ..., \succ (s_{-n}))$*, où chaque* $s_{-i} \in S_{-i}$.

Dans ce contexte, un profil de stratégies s est un équilibre de Nash d'un jeu stratégique avec préférences paramétrées G si et seulement si pour tout $i \in [1, ..., n]$*, pour tout* $s'_i \in S_i, s_i \succeq (s_{-i})s'_i$.

Il est montré dans [18] que tout jeu stratégique peut être traduit par un CP-net. Nous allons présenter ici la traduction d'un jeu vers un CP-net, la traduction réciproque étant quasiment symétrique.

Soit un jeu stratégique avec préférences paramétrées $G = (N, S1, ..., Sn, \succ (s_{-1}), ..., \succ (s_{-n}))$. Ce jeu peut être associé à un CP-net N (G) de la façon suivante :

- chaque joueur i correspond à une variable X_i,
- les éléments du domaine $D(X_i)$ de la variable X_i consistent en les stratégies du joueur i.
- $Pa(X_i) = \{X_1, ..., X_{i-1}, X_{i+1}, ..., X_n\}$ l'ensemble des parents de la variable X_i.

- pour chaque s_{-i}, la préférence conditionnelle $X_{-i} = s_{-i} :\succ (s_{-i})$ est ajoutée, où $\succ (s_{-i})$ est la relation de préférence sur l'ensemble des stratégies du joueur i si ses opposants jouent s_{-i}.

Il est montré dans [18] qu'un profil de stratégies est un équilibre de Nash du jeu G si et seulement si c'est un résultat optimal du CP-net $N(G)$.
Exprimer un jeu avec des CP-nets peut être parfois plus compact que la forme normale explicite du jeu, pourvu que les préférences de certains joueurs ne dépendent pas de tous les autres joueurs.

3.3 Les jeux et les CSPs continu

Bordeau et Pajot proposent de representer le jeu fini à n joueur par un CSP continu, puis le résoudre afin de trouver un équilibre de nash en stratégie mixte.
pour cela, ils adopté le codage suivant [19] :

- A chaque stratégie mixte $\alpha_i \in \Delta_i$ du joueur i, est associée une variable X_i du CSP, Δ_i étant défini par la relation , $i \in I$.
- Le domaine d'une variable X_i est donné par l'ensemble Δ_i des stratégies mixtes.
- Les contraintes du problem se résument au fait que chaque joueur i choisit sa stratégie mixte $\alpha_i^* \in \Delta_i$ correspondant à la variable X_i qui soit meilleur réponse au choix du reste des joueurs $\alpha_{-i}^{\sim} \in \Delta_{-i}$, i.e

$$E_i(\alpha_i^*, \alpha_{-i}^{\sim}) = max_{\alpha_i \in D_i} E_i(\alpha_i, \alpha_i^{\sim}), \forall i \in I \tag{3.1}$$

Cette contrainte de maximisation doit être exprimée sous forme de contraintes à satisfaire. Pour cela, il suffit de considérer la caractéristique suivante de l'équilibre de Nash en stratégies mixtes :

Proposition 3.1 *[26] On dit qu'une situation mixte $\alpha = (\alpha_1, ..., \alpha_n) \in \Delta$ est un équilibre de Nash pour le jeu ,si et seulement si pour tout joueur $i \in I$, on a*

$$E_i(\alpha_1, ..., \alpha_n) \geq E_i(\alpha_1, ..., \alpha_{i-1}, e_{ij}, \alpha_{i+1}, ..., \alpha_n), \forall j = \overline{1, m_i} \tag{3.2}$$

où e_{ij} est la stratégie mixte du joueur i dont les composantes sont toutes nulles, à l'exception de la j^{ieme} composante, qui est égale à 1. Une telle stratégie mixte exprime le fait que le joueur i choisit à coup sûre sa stratégie pure $s_i \in S_i$.

Notons que les résultats obtenus représentent des approximations de l'équilibre de Nash en stratégies mixtes. L'analyse des résultats obtenus révèle que la méthode des intervalles contraints est très performante pour la résolution de problèmes CSPs ayant au plus 10 variables. Par contre, en passant d'un codage à un autre, cette méthode peut mener vers des résultats inhabituels, comparés aux performances qu'elle présente en l'applicant à la

résolution de problèmes autres que ceux de la théorie des jeux. Chose que Bordeau et Pajot [26] n'arrivent pas à justifier, et pensent que ceci peut être dû à la redondance des variables du CSP continue dans toutes les contraintes du problème [26].

3.4 Les jeux stratégiques et les softs CSP

Rossi et al s'intéressent dans [34] à la possibilité de relier deux formalismes, à savoir : les jeux stratégiques et les softs CSP [1]. Ils ont étudié la relation entre les jeux stratégiques représentés sous une forme graphique et diverses classes de softs CSPs. A partir de là, il a été conclu qu'en représentant un soft CSP (SCSP) par un jeu stratégique, on ne peut déduire aucune relation entre l'espace des solutions optimales du SCSP et celui des équilibres de Nash du jeu associé.

En revanche, la modélisation d'un jeu stratégique par un soft CSP permet de prouver que toute solution optimale du SCSP est un équilibre Pareto-optimal du jeu en question. Si de plus, on rajoute des contraintes strictes [2] adéquates au SCSP, ces solutions optimales coincident avec les équilibres Nash-Pareto optimaux.

De tels résultats peuvent être exploités de différentes manières dont la plus évidente est d'exploiter les algorithmes de calculs dédiés à la résolution d'un des deux problèmes pour en résoudre l'autre [34, 11]. Par exemple, pour rechercher l'équilibre de Pareto pour un jeu donné, on représente ce jeu par un SCSP, puis on le résout.

3.5 Les jeux de contraintes

Plus récemment, une nouvelle classe de jeux, dite *jeux de contraintes*, a été définit dans [42] pour une représentation plus compacte de l'espace des profils de stratégies et des matrices d'utilités dans les jeux statiques, à l'aide des outils de la PPC.
Dans cette classe de jeux, les préférences des joueurs sont représentées par un problème de satisfaction de contraintes (CSP) ou un problème d'optimisation de contraintes (COP) dont la satisfaction (ou maximisation dans le cas d'un COP) définit une situation préférée par le joueur. Dans [42], un algorithme basé sur la recherche arborescente a été développé pour la résolution de ces jeux, son avantage est qu'il est capable de garder en mémoire les meilleures réponses déjà examinées pour éliminer tout calcul dupliqué.
Une amélioration de cet algorithme a été proposée dans [35], où les préférences des joueurs sont représentées par des contraintes globales, ce qui a permis d'apporter des améliorations en terme d'efficacité et de vitesse de calcul. En effet, ce nouveau solveur est capable de trouver des équilibres de Nash en stratégies pures pour certains problèmes, où le nombre de joueurs peut aller jusqu'à 200. Pour le cas des jeux graphiques, le solveur permet de résoudre des instances, où le nombre de joueurs peut atteindre les 2000.

1. Les Soft CSPs sont des CSPs où il existe une préférence ou priorité entre les contraintes [33]
2. On appelle " contraintes strictes " des contraintes qui doivent forcément être satisfaites.

3.6 Conclusion

A travers ce chapitre, nous avons exposé quelques travaux de littérature parmi les plus importants ayant abordé la question du calcul des équilibre en utilisant les outils de la programmation par contraintes. En se basant sur cette synthèse, nous allons proposer dans le chapitre suivant notre contribution dans ce contexte.

4

Modèle CSP pour la Représentation des Dynamiques de Recherche des Stratégies de Meilleures Réponses

4.1 Introduction

La théorie des jeux et les CSPs représentent des outils puissants pour la modélisation d'une grande classe de problèmes rencontrés dans divers domaines. De plus, ces deux théories peuvent s'appliquer l'une à l'autre. Dans ce chapitre, nous allons nous intéresser à la modélisation par CSP du problème de recherche d'un équilibre de Nash en stratégies pures, en passant par le calcul des stratégies de meilleures réponses.

4.2 Position du problème

Soit un jeu non coopératif J défini par le triplet :

$$J = \left\langle I, \{S_i\}_{i \in I}, \{f_i\}_{i \in I} \right\rangle \tag{4.1}$$

où :

- $I = \{1, .., n\}$ est un ensemble fini de n joueurs ;
- $S_i = \{s_{i1}, .., s_{im_i}\}$ est un ensemble fini qui contient les stratégies pures d'un joueur $i \in I,\ \forall i \in I$;
- $f_i(.)$ est la fonction d'utilité d'un joueur $i \in I$.

Résoudre le jeu (4.1) revient à calculer un équilibre de Nash. Cet équilibre peut être défini comme une situation du jeu où chaque joueur choisit une stratégie de meilleure réponse aux stratégies s^*_{-i} choisies par ses adversaires, i.e une issue $s^* = (s^*_1, \ldots, s^*_n) \in S$ est un équilibre de Nash du jeu (4.1) si et seulement si :

$$s^*_i \in MR(s^*_{-i}), \quad \forall i \in I \tag{4.2}$$

avec

$$MR_i(s^*_{-i}) = \left\{ \tilde{s_i} \in S_i, \quad f_i(\tilde{s_i}, s^*_{-i}) \geq f_i(s_i, s^*_{-i}), \quad \forall s_i \in S_i \right\} \tag{4.3}$$

La question qu'on se pose ici est comment peut-ont exploiter les outils de la programmation par contraintes pour représenter et résoudre le problème de recherche d'une issue $s^* = (s^*_i, s^*_{-i})$ vérifiant la relation (4.2). Dans la section suivante nous allons justement proposer un modèle CSP pour représenter le problème posé.

4.3 Le modèle proposé

Notons par

$$P = \langle X, D, C \rangle, \tag{4.4}$$

le problème de satisfaction de contraintes que nous proposons pour modéliser la question de détermination d'un équilibre de Nash en stratégies pures pour le jeu (4.1).

Dans ce modèle, les éléments du CSP(P) sont définis comme suit :

1. $X = \{X_1, \ldots, X_n\}$ est l'ensemble des variables de décision. Chaque variable X_i représente la stratégie pure choisie par un joueur $i \in I$;
2. $D = \{D_1, \ldots, D_n\}$ est l'ensemble des domaines des variables de décision. Chaque domaine D_i coincide avec l'ensemble des stratégies du joueur associé à la variable X_i, i.e $D_i = S_i,\ i = \overline{1, n}$;
3. $C = \{C_1, \ldots, C_n\}$ est l'ensemble des contraintes du problèmes. Chaque contrainte C_i est une contrainte unaire imposée sur la valeur de la variable de décision de même indice, ç-à-d sur la valeur de X_i.
   Pour formuler ces contraintes, nous allons noter par $x^* = (x^*_1, \ldots, x^*_n) = (x^*_i, x^*_{-i})$ une solution au CSP(P), avec x^*_{-i} est le vecteur solution sans la valeur de la $i^{\text{ème}}$ composante. Chaque contrainte $C_i \in C$ impose que la valeur x^*_i de chaque variable de décision soit un élément de l'ensemble $MR_i(x^*_{-i})$, défini par la relation (4.3).
   Formellement :
   $$C = \left\{ x^*_i \in MR_i(x^*_{-i}),\ i = \overline{1, n} \right\} \tag{4.5}$$
   On peut également exprimer l'ensemble des contraintes C par la relation :
   $$C = \left\{ x^*_i \in D_i / \quad x^*_i = \arg\max_{x_i \in D_i} f_i(x_i, x^*_{-i}), \quad i = \overline{1, n} \right\}, \tag{4.6}$$

ou encore par la relation :

$$C = \left\{ f_i(x_i^*, x_{-i}^*) \geq f_i(x_i, x_{-i}^*), \forall x_i \in D_i, \quad x_i \neq x_i^*, \quad \forall i = \overline{1,n} \right\}, \tag{4.7}$$

Proposition 4.1 *Toute solution au CSP* (4.4) *est un équilibre de Nash en stratégies pures pour le jeu* (4.1).

Preuve 4.1 *La démonstration de cette proposition découle directement de la formulation des contraintes imposées aux valeurs des variables de décision du CSP* (4.4). *En effet, ces contraintes imposent que chaque variable X_i associée à la stratégie d'un joueur $i \in I$ prenne une valeur qui représente une stratégie de meilleure réponse par rapport aux valeurs attribuées aux autres variables (qui correspondent aux stratégies choisies par les adversaires du joueur i).*

4.4 Illustration du modèle proposé sur le jeu du dilemme du prisonnier

Le dilemme du prisonnier (DP), énoncé en 1950 par *Albert W. Tucker* à Princeton, caractérise en théorie des jeux une situation où deux joueurs auraient intérêt à coopérer, mais où, en l'absence de communication entre les deux joueurs, chacun choisira de trahir l'autre si le jeu n'est joué qu'une fois. La raison est que si l'un coopère et que l'autre trahit, le coopérateur est fortement pénalisé. Pourtant, si les deux joueurs trahissent, le résultat leur est moins favorable que si les deux avaient choisi de coopérer.

4.4.1 Principe du dilemme du prisonnier (DP)

Tucker suppose deux prisonniers (complices d'un crime) retenus dans des cellules séparées et qui ne peuvent pas communiquer. L'autorité pénitentiaire offre à chacun des prisonniers les choix suivants :

- Si un seul des deux prisonniers dénonce l'autre, il est remis en liberté alors que le second obtient la peine maximale (10 ans) ;
- Si les deux se dénoncent entre eux, ils seront condamnés à une peine plus légère (5 ans) ;
- Si les deux refusent de dénoncer, la peine sera minimale (6 mois), faute d'éléments au dossier.

Ce problème modélise bien les questions de politique tarifaire : le concurrent qui baisse ses prix gagne des parts de marché et peut ainsi augmenter ses ventes et accroître éventuellement son bénéfice, mais si son concurrent principal en fait autant, les deux peuvent y perdre.

$f_1(s_1, s_2)$	$s_2 = T$	$s_2 = D$
$s_1 = T$	-6	-10
$s_1 = D$	0	-5

$f_2(s_1, s_2)$	$s_2 = T$	$s_2 = D$
$s_1 = T$	-6	0
$s_1 = D$	-10	-5

TABLE 4.1 – Fonctions d'utilités des joueurs

4.4.2 Formulation classique du jeu du dilemme du prisonnier

Le problème du dilemme du prisonnier décrit dans la section 4.4.1 peut être représenté par le jeu stratégique suivant :

$$J_D = \langle I_D, \{S_i\}_{i \in I_D}, \{f_i\}_{i \in I_D} \rangle \tag{4.8}$$

où

- $I_D = \{1, 2\}$ est l'ensemble des deux prisonniers qui représentent les joueurs ;
- $S_i = \{T, D\}, \forall i \in I_D$ est l'ensemble des stratégies pures des deux joueurs, telle que si $s_i = T$ cela signifie que le joueur i se tait, sinon si $s_i = D$ alors le joueurs i dénonce son adversaire ;
- Les fonctions d'utilités des deux joueurs sont décrites par le tableau 4.1.

4.4.3 Formulation du CSP associé au jeu du dilemme du prisonnier

Le jeu (4.8) étant un jeu noncoopératif, le concept de solution le plus adéquat est celui de l'équilibre de Nash lorsque ce dernier existe. Nous proposons ici d'adapter le modèle CSP défini par la relation (4.4) pour calculer un équilibre de Nash pour le jeu du dilemme du prisonnier.

Notons par

$$P_D = \langle X_D, D_D, C_D \rangle, \tag{4.9}$$

le CSP associé au jeu (4.8), tel que :

- $X_D = \{X_1, X_2\}$ est l'ensemble des variable de décision des deux prisonniers ;
- $D_D = \{D_1, D_2\}$ avec $D_1 = D_2 = S_1 = S_2$ est l'ensemble des valeurs possibles pour les variables de X_D ;
- Chaque solution $x^* = (x_1^*, x_2^*)$ doit vérifier les contraintes du CSP(P_D), qui sont formulées comme suit :

$$f_1(x_1^*, x_2^*) \geq f_1(x_1, x_2^*), \quad \forall x_1 \in D_1, x_1 \neq x_1^* \tag{4.10}$$
$$f_2(x_1^*, x_2^*) \geq f_2(x_1^*, x_2), \quad \forall x_2 \in D_2, x_2 \neq x_2^* \tag{4.11}$$

4.4.4 Résolution du CSP(P_D)

Pour résoudre le CSP(P_D), nous nous retrouvons face à deux questions primordiales. En effet :

1. Comment doit-on ordonner les variables à instancier ?
2. Comment choisir efficacement les valeurs à affecter aux différentes variables ?

Pour notre cas, nous allons adopter la méthode par retours-arrière, en instanciant les variables par ordre séquentiel (i.e x_1 puis x_2), le même ordre sera considéré pour le choix des valeurs à affecter. On obtient alors les étapes suivantes :

– **Posons** $x_1^* = T$ **:** en remplaçant la variable x_1^* par sa valeur dans les contraintes (4.10) et (4.11), on aura à vérifier les contraintes suivantes :

$$f_1(T, x_2^*) \geq f_1(D, x_2^*) \tag{4.12}$$
$$f_2(T, x_2^*) \geq f_2(T, x_2), \quad \forall x_2 \in D_2, x_2 \neq x_2^* \tag{4.13}$$

- **Si** $x_2^* = T$ **:** en remplaçant x_2^* par sa valeur dans les contraintes (4.12) et (4.13), on aura à vérifier les contraintes suivantes :

$$f_1(T, T) \geq f_1(D, T) \tag{4.14}$$
$$f_2(T, T) \geq f_2(T, D) \tag{4.15}$$

  La contrainte (4.14) n'est pas vérifiée car $-6 \ngeq 0$, cependant la contrainte (4.15) n'est pas vérifié ($-6 \ngeq -2$), d'où la valeur $x_2^* = T$ est rejettée.

- **Si** $x_2^* = D$ **:** en remplaçant x_2^* par sa valeur dans les contraintes (4.12) et (4.13), on aura à vérifier les contraintes suivantes :

$$f_1(T, D) \geq f_1(D, D) \tag{4.16}$$
$$f_2(T, D) \geq f_2(T, T) \tag{4.17}$$

  La contrainte (4.16) n'est pas vérifiée car $-10 \ngeq -5$.

Par conséquent, la valeur $x_2^* = D$ est également rejettée. A ce niveau un retour en arrière pour modifier la valeur de la variable précédemment instanciée s'impose. On va alors choisir une nouvelle valeur pour x_1^*.

– **Posons** $x_1^* = D$ **:** en remplaçant la variable x_1^* par sa valeur dans les contraintes (4.10) et (4.11), on aura à vérifier les contraintes suivantes :

$$f_1(D, x_2^*) \geq f_1(T, x_2^*) \tag{4.18}$$
$$f_2(D, x_2^*) \geq f_2(D, x_2), \quad \forall x_2 \in D_2, x_2 \neq x_2^* \tag{4.19}$$

- **Si** $x_2^* = T$ **:** en remplaçant x_2^* par sa valeur dans les contraintes (4.18) et (4.19), on aura à vérifier les contraintes suivantes :

$$f_1(D, T) \geq f_1(T, T) \tag{4.20}$$
$$f_2(D, T) \geq f_2(D, D) \tag{4.21}$$

  La contrainte (4.20) est vérifiée car $0 > -6$, cependant la contrainte (4.21) n'est pas vérifié ($-10 \ngeq -5$), d'où la valeur $x_2^* = T$ est rejettée.

- **Si** $x_2^* = D$ **:** en remplaçant x_2^* par sa valeur dans les contraintes (4.18) et (4.19), on aura à vérifier les contraintes suivantes :

$$f_1(D,D) \geq f_1(T,D) \tag{4.22}$$
$$f_2(D,D) \geq f_2(D,T) \tag{4.23}$$

  La contrainte (4.22) est vérifiée car $-5 > -10$, de même pour la contrainte (4.23).

Finalement, la solution $x^* = (D, D)$ vérifie toutes les contraintes du CSP(P_D), cette solution est un équilibre de Nash en stratégies pures du jeu (4.8). En effet, si l'on revient sur le tableau 4.1 des fonctions d'utilité des joueurs, nous constatons qu'étant dans la situation $s^* = (D, D)$, si n'importe quel joueur décide de dévier unilatéralement de sa stratégie, celui-ci va diminuer la valeur de son gain, ce qui est par définition une situation d'équilibre de Nash.

4.5 Illustration du modèle proposé sur un jeu sous forme graphique

Supposons que l'interaction entre les joueurs dans le jeu (4.1) n'est pas totale, ç'est-à-dire qu'un joueur $i \in I$ peut être en interaction qu'avec un sous ensemble de joueurs, dits voisins de i et noté par $N(i) \subset I$. Dans ce cas là, le jeu (4.1) peut être représenté sous une forme plus compacte en utilisant les jeux graphiques.

Notation :

Dans les sections suivantes, nous aurons besoin de définir un certain nombre de notations que voici :

- ▷ $N(i) \subset I$, l'ensemble des voisins du joueur $i \in I$;
- ▷ $S^{N(i)} = \prod_{j \in N(i)} S_j$ l'ensemble des combinaisons possibles des stratégies choisies par les voisins d'un joueur $i \in I$;
- ▷ $s_i \in S_i$, la stratégie choisie par le joueurs i lorsque ses voisins jouent une combinaison $s^{N(i)} \in S^{N(i)}$;
- ▷ f_i est la fonction permettant d'évaluer le gain d'un joueur i, lorsque celui-ci choisit la stratégie s_i, face à la combinaison de stratégies $s^{N(i)} \in S^{N(i)}$ choisies par ses voisins ;
- ▷ m_i est le nombre de stratégies pures du joueur $i \in I$.

4.5.1 Formulation du jeu (4.1) sous forme graphique

Soit J^G le jeu graphique associé au jeu stratégique (4.1), défini par :

$$J^G = \langle Gr, \{S_i\}_{i \in I}, M^G \rangle, \tag{4.24}$$

où :

1. $Gr = (V, E)$ est un graphe non orienté tel que :
   - ▷ $V = \{V_1, \ldots, V_n\}$ est l'ensemble des sommets du graphe. Chaque sommet $V_i \in V$ correspond à un joueur $i \in I$;
   - ▷ $E = \{a_1, a_2, ..., a_d\}$ est l'ensemble des arêtes du graphe Gr, tel qu'il existe une arête $a_l = (v_i, v_j)$ si et seulement si les choix du joueur $i \in I$ influent sur les gains du joueur $j \in I$ et inversement.
2. $S_i = \{s_{i1}, \ldots, s_{im_i}\}$ est l'ensemble des stratégies pures du joueur $i \in I$;
3. $M^G = \{M_1^G, M_2^G, ..., M_n^G\}$ est l'ensemble des matrices de gains des n joueurs, où $M_i^G = (m_{l_i k_i}^G)$ est une $m_i \times |S^{N(i)}|$ matrice, telle que :

$$m_{l_i k_i}^G = f_i(s_{l_i}, s_{k_i}^{N(i)}), \forall i \in I. \tag{4.25}$$

   En supposant que les ensembles S_i et $S^{N(i)}$ sont ordonnés pour tout $i \in I$, on aura :
   - ▷ s_{l_i} est la $l^{\text{ième}}$ stratégie pure du joueur $i \in I$;
   - ▷ $s_{k_i}^{N(i)}$ est la $k^{\text{ième}}$ combinaison des stratégies choisies par les voisin du $i^{\text{ième}}$ joueur .

4.5.2 Formulation du CSP associé au jeu graphique (4.24)

Le CSP associé au jeu graphique (4.24) est défini par le triplet :

$$P^G = \langle L, D^G, C^G \rangle \tag{4.26}$$

tels que :

1. $L = \{L_1, \ldots, L_n\}$ est l'ensemble des variables de décision où chaque variable L_i désigne l'indice de la stratégie pure choisie par un joueur $i \in I$ dans l'ensemble S_i ;
2. $D^G = \{D_1^G, \ldots, D_n^G\}$ où $D_i^G = \{1, \ldots, m_i\}$ est le domaine de la variable L_i, $i \in I$, m_i est le nombre de stratégies pures du joueur i ;
3. En notons par $l^* = (l_i^*, k_i^*)$ une solution au CSP P^G, on définit les contraintes du problème par l'ensemble :

$$C^G = \left\{ m_{l_i^* k_i^*}^G \geq m_{l_i k_i^*}^G,\ \forall l_i \in D_i^G,\ l_i \neq l_i^*,\ \forall i \in I \right\}, \tag{4.27}$$

   avec k_i^* est l'indice de la combinaison des stratégies choisies par les voisins du joueurs $i \in I$ dans l'ensemble $S^{N(i)}$.

Exemple de jeu graphique

Considérons le jeu fini sous forme normale suivant :

$$J = \langle I, \{S_i\}_{i \in I}, \{f_i\}_{i \in I} \rangle \tag{4.28}$$

où

- ◇ $I = \{1, 2, 3, 4, 5\}$;
- ◇ $S_i = \{-1, 0, 1\}, \forall i \in I$;
- ◇ Les gains des joueurs en fonction de la situation du jeu $s = (s_1, s_2, ..., s_5)$ sont donnés par les fonctions :
  - ○ $f_1(s) = s_1 + 2s_2 - s_3$;
  - ○ $f_2(s) = s_1 + 3s_2 - 5s_5$;
  - ○ $f_3(s) = s_1 + s_3$;
  - ○ $f_4(s) = s_4 - s_5$;
  - ○ $f_5(s) = s_2 + s_4 + s_5$.

Construction de la forme graphique de jeu (4.28)

Soit P^G la forme graphique du jeu (4.28), tel que :

$$P^G = \langle Gr = (V, E), \{S_i\}_{i \in I}, M^G \rangle, \tag{4.29}$$

Avec :

1. $V = \{1, 2, 3, 4, 5\}$;
2. A partir des fonctions de gains des joueur dans le jeu (4.28), nous déduisons l'ensemble des voisins $N(i)$ pour chaque joueur i :
   - ○ $N(1) = \{2, 3\}$;
   - ○ $N(2) = \{1, 5\}$;
   - ○ $N(3) = \{1\}$;
   - ○ $N(4) = \{5\}$;
   - ○ $N(5) = \{2, 4\}$.

   On obtient alors l'ensemble des arêtes $E = \{a_1, a_2, a_3, a_4\}$. A ce niveau, nous pouvons tracer le graphe représentatif du jeu (4.28) comme suit :

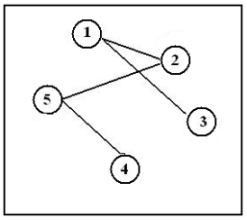

FIGURE 4.1 – Représentation graphique du jeu (4.28).

Le calcul des matrices des gains des joueurs se fait avec la formule (4.25). la lecture d'un élément sur la matrice si on prend par exemple $M_1^G(3,2) = -1$ est comme suit : -1 c'est le gain du joueur 1 lorsque celui-ci choisit sa $3^{\text{ième}}$ stratégie (s_{13}) et que ces adversaires choisissent la $2^{\text{ième}}$ combinaison de l'ensemble $S^{N(1)}$ (ie : ils choisissent $(-1,0)$).

3. Calcul des matrices de gains M_i^G :

• Gain du joueur 1 :

$S^{N(1)} = \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)\}$, tel que $S^{N(1)}$ est l'ensemble des combinaisons des stratégies du voisinage du joueur 1.

$S_1 = \{-1,0,1\}$ est l'ensemble des stratégies du joueur 1.

la matrice M_1^G est défini comme suit :

$$M_1^G = \begin{pmatrix} -2 & -3 & -4 & 0 & -1 & -2 & 2 & 1 & 0 \\ -1 & -2 & -3 & 1 & 0 & -1 & 3 & 2 & 1 \\ 0 & -1 & -2 & 2 & 1 & 0 & 4 & 3 & 2 \end{pmatrix} \tag{4.30}$$

• Gain du joueur 2 :

$S^{N(2)} = \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)\}$, tel que $S^{N(2)}$ est l'ensemble des combinaisons des stratégies du voisinage du joueur 2.

$S_2 = \{-1,0,1\}$ est l'ensemble des stratégies du joueur 2.

la matrice M_2^G est défini comme suit :

$$M_2^G = \begin{pmatrix} 1 & -4 & -9 & 2 & -3 & -8 & 3 & -2 & -7 \\ 4 & -1 & -6 & 5 & 0 & -5 & 6 & 1 & -4 \\ 7 & 2 & -3 & 8 & 3 & -2 & 9 & 4 & 1 \end{pmatrix} \tag{4.31}$$

• Gain du joueur 3 :

$S^{N(3)} = \{-1,0,1\}$, tel que $S^{N(3)}$ est l'ensemble des combinaisons des stratégies du voisinage du joueur 3.

$S_1 = \{-1,0,1\}$ est l'ensemble des stratégies du joueur 3.

la matrice M_3^G est défini comme suit :

$$M_3^G = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} \tag{4.32}$$

• Gain du joueur 4 :

$S^{N(4)} = \{-1,0,1\}$, tel que $S^{N(4)}$ est l'ensemble des combinaisons des stratégies du voisinage du joueur 4.

$S_4 = \{-1,0,1\}$ est l'ensemble des stratégies du joueur 4.

la matrice M_4^G est défini comme suit :

$$M_4^G = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix} \tag{4.33}$$

• Gain du joueur 5 :

$S^{N(5)} = \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)\}$, tel que $S^{N(5)}$ est l'ensemble des combinaisons des stratégies du voisinage du joueur 5.
$S_5 = \{-1,0,1\}$ est l'ensemble des stratégies du joueur 5.
la matrice M_5^G est défini comme suit :

$$M_5^G = \begin{pmatrix} -3 & -2 & -1 & -2 & -1 & 0 & -1 & 0 & 1 \\ -2 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 2 \\ -1 & 0 & 1 & 0 & 1 & 2 & 1 & 2 & 3 \end{pmatrix} \tag{4.34}$$

Formulation du CSP associé au jeu graphique (4.29)

$$P^G = \langle L, D^G, C^G \rangle \tag{4.35}$$

◇ $L = \{L_1, L_2, L_3, L_4, L_5\}$ est l'ensemble des variables de décision du joueur i tel que $i \in I$;

◇ $D_i^G = \{1,2,3\}; \forall i = \overline{1,5}$ est l'ensemble des valeurs possibles pour les variables de L ;

◇ Chaque solution $l^* = (l_1^*, l_2^*, l_3^*, l_4^*, l_5^*)$ doit vérifier les contraintes du CSP(P^G), qui sont formulées comme suit :

$$m^G_{l_1^* k_1^*} = f_1(s_{l_1^*}, s^{N(1)}_{k_1^*}) \;\geq\; m^G_{l_1 k_1^*} = f_1(s_{l_1}, s^{N(1)}_{k_1^*}), \quad \forall l_1 \in D_1^G, l_1 \neq l_1^* \tag{4.36}$$

$$m^G_{l_2^* k_2^*} = f_2(s_{l_2^*}, s^{N(2)}_{k_2^*}) \;\geq\; m^G_{l_2 k_2^*} = f_2(s_{l_2}, s^{N(2)}_{k_2^*}), \quad \forall l_2 \in D_2^G, l_2 \neq l_2^* \tag{4.37}$$

$$m^G_{l_3^* k_3^*} = f_3(s_{l_3^*}, s^{N(3)}_{k_3^*}) \;\geq\; m^G_{l_3 k_3^*} = f_3(s_{l_3}, s^{N(3)}_{k_3^*}), \quad \forall l_3 \in D_3^G, l_3 \neq l_3^* \tag{4.38}$$

$$m^G_{l_4^* k_4^*} = f_4(s_{l_4^*}, s^{N(4)}_{k_4^*}) \;\geq\; m^G_{l_4 k_4^*} = f_4(s_{l_4}, s^{N(4)}_{k_4^*}), \quad \forall l_4 \in D_4^G, l_4 \neq l_4^* \tag{4.39}$$

$$m^G_{l_5^* k_5^*} = f_5(s_{l_5^*}, s^{N(5)}_{k_5^*}) \;\geq\; m^G_{l_5 k_5^*} = f_5(s_{l_5}, s^{N(5)}_{k_5^*}), \quad \forall l_5 \in D_5^G, l_5 \neq l_5^* \tag{4.40}$$

4.5.3 Résolution du CSP(P^G)

Pour notre cas, nous allons adopter la méthode par retours-arrière, en instanciant les variables par ordre séquentiel (i.e l_1, l_2,l_3,l_4, l_5), le même ordre sera considéré pour le choix des valeurs à affecter. On obtient alors les étapes suivantes :

– **Posons** $l_1^* = 1$ **:** on doit instancier ses voisins $N(1) = \{2,3\}$, commencons par instancier la variable l_2

• **Posons** $l_2^* = 1$ **:** on remplaçant l_1^* et l_2^* par leurs valeurs, on aura à vérifier la contrainte (4.37).

La contrainte (4.37) n'est pas vérifié car :

$$M_2^G(s_{21}, s_{11}^{N(2)}) < M_2^G(s_{23}, s_{11}^{N(2)}) \quad (ie : 1 < 7) \tag{4.41}$$
$$M_2^G(s_{21}, s_{12}^{N(2)}) < M_2^G(s_{23}, s_{12}^{N(2)}) \quad (ie : -4 < 2) \tag{4.42}$$
$$M_2^G(s_{21}, s_{13}^{N(2)}) < M_2^G(s_{23}, s_{13}^{N(2)}) \quad (ie : -9 < -3) \tag{4.43}$$

- **Posons** $l_2^* = 2$ **:** on remplaçant l_1^* et l_2^* par leurs valeurs, on aura à vérifier la contrainte (4.37).
  La contrainte (4.37) n'est pas vérifié car :

$$M_2^G(s_{22}, s_{11}^{N(2)}) < M_2^G(s_{23}, s_{11}^{N(2)}) \quad (ie : 4 < 7) \tag{4.44}$$
$$M_2^G(s_{22}, s_{12}^{N(2)}) < M_2^G(s_{23}, s_{12}^{N(2)}) \quad (ie : -1 < 2) \tag{4.45}$$
$$M_2^G(s_{22}, s_{13}^{N(2)}) < M_2^G(s_{23}, s_{13}^{N(2)}) \quad (ie : -6 < -3) \tag{4.46}$$

- **Posons** $l_2^* = 3$ **:** on remplaçant l_1^* et l_2^* par leurs valeurs, on aura à vérifier la contrainte (4.37).
  La contrainte (4.37) est vérifié, donc $l_2^* = 3$. Dans ce cas là, on doit instancier ses voisins $N(2) = \{1, 5\}$. Pour la variable l_1 c'est déja vérifié, donc on aura à instancier l_5.

  - **Posons** $l_5^* = 1$ **:** on remplaçant l_2^* et l_5^* par leurs valeurs, on aura à vérifier la contrainte (4.40).
    La contrainte (4.40) n'est pas vérifié car :

$$M_5^G(s_{51}, s_{57}^{N(5)}) < M_5^G(s_{53}, s_{57}^{N(5)}) \quad (ie : -1 < 1) \tag{4.47}$$
$$M_5^G(s_{51}, s_{58}^{N(5)}) < M_5^G(s_{53}, s_{58}^{N(5)}) \quad (ie : 0 < 2) \tag{4.48}$$
$$M_5^G(s_{51}, s_{59}^{N(5)}) < M_5^G(s_{53}, s_{59}^{N(5)}) \quad (ie : 1 < 3) \tag{4.49}$$

    Puisque c'est pas vérifié, on suppose une autre valeur pour l_5^*

  - **Posons** $l_5^* = 2$ **:** on remplaçant l_2^* et l_5^* par leurs valeurs, on aura à vérifier la contrainte (4.40).
    La contrainte (4.40) n'est pas vérifié car :

$$M_5^G(s_{52}, s_{57}^{N(5)}) < M_5^G(s_{53}, s_{57}^{N(5)}) \quad (ie : 0 < 1) \tag{4.50}$$
$$M_5^G(s_{52}, s_{58}^{N(5)}) < M_5^G(s_{53}, s_{58}^{N(5)}) \quad (ie : 1 < 2) \tag{4.51}$$
$$M_5^G(s_{52}, s_{59}^{N(5)}) < M_5^G(s_{53}, s_{59}^{N(5)}) \quad (ie : 2 < 3) \tag{4.52}$$

    puisque c'est pas vérifié on suppose une autre valeur pour l_5^*

- **Posons** $l_5^* = 3$ **:** on remplaçant l_2^* et l_5^* par leurs valeurs, on aura à vérifier la contrainte (4.40).
  La contrainte (4.40) est vérifié, Danc $l_5^* = 3$. Dans ce cas là, on doit instancier ses voisins $N(5) = \{2, 4\}$. Pour la variable l_2 c'est déja vérifié, donc on aura à instancier l_4.

  - **Posons** $l_4^* = 1$ **:** on remplaçant l_5^* et l_4^* par leurs valeurs, on aura à vérifier la contrainte (4.39).
    La contrainte (4.39) n'est pas vérifié car :

$$M_4^G(s_{41}, s_{43}^{N(4)}) < M_4^G(s_{43}, s_{43}^{N(4)}) \quad (ie : -2 < 0) \tag{4.53}$$

    puisque c'est pas vérifié on suppose une autre valeur pour l_4^*

  - **Posons** $l_4^* = 2$ **:** on remplaçant l_5^* et l_4^* par leurs valeurs, on aura à vérifier la contrainte (4.39).
    La contrainte (4.39) n'est pas vérifié car :

$$M_4^G(s_{42}, s_{43}^{N(4)}) < M_4^G(s_{43}, s_{43}^{N(4)}) \quad (ie : -1 < 0) \tag{4.54}$$

    puisque c'est pas vérifié on suppose une autre valeur pour l_4^*

  - **Posons** $l_4^* = 3$ **:** on remplaçant l_5^* et l_4^* par leurs valeurs, on aura à vérifier la contrainte (4.39).
    La contrainte (4.39) est vérifié. Danc $l_4^* = 3$. Dans ce cas là, on doit instancier ses voisins $N(4) = \{5\}$. Pour la variable l_5 c'est déja vérifié.

maintenant on doit instancier l'autre voisin de l_1 qui est la variable l_3.

- **Posons** $l_3^* = 1$ **:** on remplaçant l_1^* et l_3^* par leurs valeurs, on aura à vérifier la contrainte (4.38).
  La contrainte (4.38) n'est pas vérifié car :

$$M_3^G(s_{31}, s_{33}^{N(3)}) < M_3^G(s_{33}, s_{33}^{N(3)}) \quad (ie : 0 < 2) \tag{4.55}$$

  puisque c'est pas vérifié on suppose une autre valeur pour l_3^*

- **Posons** $l_3^* = 2$ **:** on remplaçant l_1^* et l_3^* par leurs valeurs, on aura à vérifier la contrainte (4.38).
  La contrainte (4.38) n'est pas vérifié car :

$$M_3^G(s_{32}, s_{33}^{N(3)}) < M_3^G(s_{33}, s_{33}^{N(3)}) \quad (ie : 1 < 2) \tag{4.56}$$

puisque c'est pas vérifié on suppose une autre valeur pour l_3^*

- **Posons** $l_3^* = 3$ **:** on remplaçant l_1^* et l_3^* par leurs valeurs, on aura à vérifier la contrainte (4.38).
  La contrainte (4.38) est vérifié, donc $l_3^* = 3$. La variable l_3 n'a qu'un seul voisin qui est l_1.

Donc on aura à vérifier si l'issu $l^* = (1, 3, 3, 3, 3)$ satisfait la contrainte (4.36). Ce qui n'est pas vérifié car : on remplaçant l_1^*, l_2^* et l^*3 par leurs valeurs dans la contrainte (4.36) on aura :

$$M_1^G(s_{11}, s_{19}^{N(1)}) < M_1^G(s_{19}, s_{19}^{N(1)}) \quad (ie : 0 < 2) \tag{4.57}$$

D'où l'issu $l^* = (1, 3, 3, 3, 3) \notin C^G$.

Dans ce cas là, on doit supposer une autre valeur pour l_1^* qui vérifié la contrainte (4.36).

– **Posons** $l_1^* = 2$ **:** On remplaçant l_1^*, l_2^* et l^*3 par leurs valeurs dans la contrainte (4.36) on aura :

$$M_1^G(s_{12}, s_{19}^{N(1)}) < M_1^G(s_{19}, s_{19}^{N(1)}) \quad (ie : 1 < 2) \tag{4.58}$$

La contrainte (4.36) n'est pas vérifié, d'où l'issu $l^* = (2, 3, 3, 3, 3) \notin C^G$.

Dans ce cas là, on doit supposer une autre valeur pour l_1^* qui vérifié la contrainte (4.36).

– **Posons** $l_1^* = 3$ **:** On remplaçant l_1^*, l_2^* et l^*3 par leurs valeurs dans la contrainte (4.36) on aura :
La contrainte (4.36) est vérifié, donc $l_1^* = 3$.
D'où l'issu $l^* = (3, 3, 3, 3, 3) \in C^G$. donc c'est une solution pour le CSP (4.35). d'autre part, cet issu est l'équilibre de Nash du jeu sous forme graphique (4.29).

4.6 Conclusion

Dans ce chapitre, nous avons présenté la formulation des CSP associé à un jeu sous forme normale et à un jeu graphique, puis les résoudre en résolvant le problème de recherche d'une stratégie de meilleure réponse, toute en utilisant les outils de la programmation par contraintes.

Conclusion Générale

Le travail entrepris dans le cadre de ce mémoire porte essentiellement sur le problème de calcul de l'équilibre de Nash, qui reste l'un des concepts de solution les plus utilisés dans l'application de la théorie jeux. Cependant, la recherche de cet équilibre peut s'avèrer diffcile. En effet, les algorithmes dédiés au calcul de l'équilibre de Nash sont souvent inexploitables lorsqu'il s'agit de problèmes dont le nombre de joueurs et/ ou de leurs stratégies est grand.

Nous nous somme intéressés d'une part, aux problèmes de satisfaction de contraintes classiques (CSP) ainsi qu'aux notions principales qui leurs sont liées. Nous avons présenté le processus de modélisation d'un problème sous forme d'un CSP et quelques approches classiques permettant de les résoudre. D'autre part, nous avons défini ce qu'est la théorie des jeux et ses concepts de base, puis nous avons résumé quelques concepts de solution pour un jeu.

La théorie des jeux et la programmation par contraintes étant deux théorie pouvant intervenir dans la modélisation d'une grande variété de problèmes combinatoires, la possibilité de lier ces deux théories n'est pas à exclure. Certains travaux novateurs se sont intéressés à l'étude de ces liens pouvant présenter un intérêt calculatoire considérable.
Par conséquent, nous avons consacré une partie de ce mémoire à la présentation d'une synthèse bibliographie des travaux étudiant juste la modélisation des problème de la théorie des jeux par le problème de satisfaction de contraintes (CSP), et exploiter les concepts de solution d'un CSP dans la résolution et la recherche d'un équilibre du jeu.

Notre contribution consiste en la proposition d'un modèle CSP pour la modélisation d'un jeu sous forme normale et définir l'ensemble des stratégies de meilleures réponses à l'aide des outils de la programmation par contraintes. Puis, nous avons pris comme exemple le jeu de dilemme de prisonier, et un jeu graphique dont la modélisation et résolution est faite par le CSP.

En guise de perspectives, nous envisageons de finaliser ce travail en intégrant une partie expérimentale, où nous allons intégrer notre modèle dans un solveur CSP choisi parmi les nombreux solveurs open-source existants. Puis d'étudier les différentes classes de jeux auxquelles cette approche peut s'avérer efficace.

Bibliographie

[1] Jeux classiques en théorie algorithmique des jeux, chapitre 5. *Iri.fr/ Jcohen/documents/enseignement/chap5.edf.*

[2] Robert Adkins. Algorithmic game theory. *Faculty advisor : david moun*, 2015.

[3] Anthony Palmieri Arnaud Lalouet. Recherche d'heuristique d'équilibre de nash : Quelques résultats préliminaires pour les constraint games. *Treizièmes Journées Francophones de Programmation par Contraintes, JFPC 2017, Montreuil sur Mer, France. hal-01628436*, Jun 2017.

[4] Nicolas berger. Modélisation et résolution en programmation par contraintes de problèmes mixtes/continues/ discrets de satisfaction de contraintes et d'optimisation. *Université de Nanttes*, 2010.

[5] C. Bessiere and J Régin. Mac and combined heuristics : two reasons to forsake fc (and cbj ?) on hard problems. *In Proceedings of CP'96, pages 61–75.*, 1996.

[6] Kahina Bouchama. Le clustering par la théorie des jeux et la programmation par contraintes. *Université de Béjaia*, 2020.

[7] S. A. Cook. The complexity of theorem-proving procedures. *In Proceedings of the Third Annual ACM Symposium on Theory of Computing, pages 151–158, New York (USA), Association for Computing Machinery*, 1971.

[8] T. Dac-Huy. Sectorisation contrainte de l'espace aérien, thèse de doctorat. *Université de Technologie de Compiègne*, 2004.

[9] N.Berline et al. Introduction à la théorie des jeux répétés. *Livre. les editions de l'école polytechnique*, 2007.

[10] Daphne KOLLER et Brian MILCH. Multi-agent influence diagrams for representing and solving games. games and economic behavior,. *Full version of paper in IJCAI '01.,*, 2003.

[11] E.Vareilles. Conception et approches par propagation de contraintes : La mise en oeuvre d'un outil d'aide intéractif thèse de doctorat,. *institut national polytechnique de Toulouse.*, 2005.

[12] D. Frost and R Dechter. In search of the best constraint satisfaction search. *In AAAI, pages 301–306.*, 1994.

[13] Gianluigi GRECO et Francesco SCARCELLO. Georg GOTTLOB. Pure nash equilibria : Hard and asy games.,. *Journal of Artificial Intelligence Research, 24 :357–406,*, 2005.

[14] Paul M.lepelley D.etsmaoui H.cemoi. Introduction à la théorie des jeux : les jeux non cooperatifs. *Université de REUNION.*

[15] Sébastien Hémon. Théorie des jeux. *LRDE-EPITA*, 2006.

[16] Min jiang. Finding pure nash equilibrium of graphical game via constraints satisfaction approach. *Wuhan University,wuhan hubei 430072 china .*, 2007.

[17] David S Johnson and Michael R Garey. Computers and intractability : A guide to the theory of np-completeness. *WH Freeman*, 1979.

[18] Fancesca ROSSI et Kristen Brent VENABLE. Krzysztof R. APT. Cp-nets and nash equilibria. *Dans Proceedings of Third International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS'05), Singapore,*, 2005.

[19] Brice bajot Lucas bordeaux. Computing equilibria using interval constraints. *LNAI no. 3419,157-171 .*, 2005.

[20] Ernst. ludwing von thadden. cour : introduction à la théorie des jeux : théorie, applications, problemes. 2004.

[21] Arnaud malapert. techniques d'ordannancement d'atelier et de fournées basées sur la programmation par contraintes. 2011.

[22] Arnaud Malapert. Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes. *PhD thesis*, 2011.

[23] Michael L. LITTMAN et Satinder SINGH. Michael KEARNS. Graphical models for game theory.,. *Dans Uncertainty in Artificial Intelligence (UAI'01),*, 2001.

[24] Guoqing wu Min jiang, Changle zahou and Fan zhang. A csp-based approach for solving parity game. *Depatrement of cognitive science, Xiamen university, fujian chaina.*, 2008.

[25] Johnston M. Philips A. Minton, S. and P Laird. Minimizing conflicts : a heuristic repair method for constraint-satisfaction and scheduling problems. *Artificial Intelligence, 58(1-3) :161–205.*, 1992.

[26] M.J.Osborne. An introduction to game théorie,. *Oxford university press.*, 2000.

[27] Boucenna mohamed lamine. Coopération dans les réseaux ad hoc par application de la théorie des jeux. *Université constantine 1*, 2014.

[28] M.S.Radjef. Cours sur la théorie des jeux et l'optimisation multicritère, 1ère année post-graduation. *Université de bejaia .*, 2010.

[29] M.S.Radjef. La théorie des jeux non coopératif, cours master 2. *Departement recherche opérationelle, Université de Béjaia*, 2014.

[30] Pierfrancesco LA MURA. Game networks,. *Dans Uncertainty in Artificial Intelligence (UAI'01),*, 2000.

[31] neumann. j.v ; morgenstern. o. Theory of games and economic behavior. *Princeton University Press*, 1944.

[32] Thi-Van-Anh Nguyen. Constraint games : Modeling and solving games with constraint. *Université de Caen*, 2014.

[33] Lakmeche omar. Replique stratégique au sein d'un oligopole approche par la théorie des jeux. *Université d'oran*, 2009/2010.

[34] O.Roussel. Une autre conversion de sat vers csp. *Actes JFPC .*, 2005.

[35] Anthony Palmieri and Arnaud Lallouet. Constraint games revisited. *IJCAI*, 2017.

[36] L. Paris. Approches pour les problèmes sat et csp : ensembles strong backdoor, voisinage consistant et forme normale généralisée, thèse de doctorat. *Université de Provence*, 2007.

[37] Belaid Saad. Intégration des problèmes de satisfaction des contraintes distribués et sécurisés dans les systemes d'aide à la décision à base de connaissance. *Ecole doctorale laem lorraine*, 2010.

[38] L. Sais. De la résolution du problème sat à la résolution de problèmes autour de sat, thèse d'hdr. *Universit´e d'Artois*, 2000.

[39] Oliver Schulte. Intdoduction to computational game theory, matrix games and nash equilibrium. *Simon Fraser university CMPT882.*

[40] Leyton Brown. K. Shoham.Y. Essentials of game theory. *Morgan and Colypool*, 2008.

[41] Arnaud Lalouet Thi-Van-Anh Nguyen. A complete solver for constraint games. *Université de Caen, GREYC, Campus Cote de Nacre, Boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex, France.*

[42] Arnaud Lalouet Thi-Van-Anh Nguyen. Constraint games : framework and local search solver. *GREYC, Université de Caen Basse-Normandie Caen, France*, 2013.

[43] Fudenberg.D ; Tirole.J. Game theory. *M.I.T Press cambridge*, 1991.

[44] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. *In Eighteenth national conference on Artificial intelligence, pages 345–351 American Association for Artificial Intelligence,*, 2002.

[45] Julien Vion. Contributions à la résolution générique des problèmes de satisfaction de contraintes. *PhD thesis*, 2007.

[46] Bakhri yassine. La théorie des jeux. *Université sidi mohamed ben abdellah*, 2015.

Résumé

L'équilibre de Nash est le concept de solution le plus connu et le plus étudié en théorie des jeux non coopératifs. Cependant, le calcul de cet équilibre est connu pour être un problème NP difficile. Ces dernières années, on recense quelques travaux novateurs ayant abordé la question du calcul de cet équilibre en utilisant les outils de la programmation par contraintes. Par conséquent, l'objectif fixé pour ce travail est de présenter dans un premier temps une synthèse de ces travaux, puis par la suite, exploiter la définition de l'équilibre de Nash en tant que situation du jeu où chaque joueur choisit une stratégie de meilleure réponse pour représenter le problème de calcul d'un équilibre de Nash en stratégies pures sous forme d'un problème de satisfaction de contraintes. Ce modèle sera illustré sur quelques classes particulières de jeux.

Mot clés : Théorie des jeux, Problème de satisfaction de contraintes(CSP), Équilibre de Nash, Stratégies de meilleure réponse.

Abstract

Nash's equilibrium is the best-known and most studied solution concept in non-cooperative game theory. However, calculating this equilibrium is known to be a difficult NP problem. In recent years, we have identified some innovative works that have addressed the question of calculating this equilibrium using the tools of constraint programming. Consequently, the objective set for this work is to first present a synthesis of this work, then subsequently, to exploit the definition of Nash equilibrium as a game situation where each player chooses a strategy of best answer to represent the problem of calculating a Nash equilibrium in pure strategies in the form of a constraint satisfaction problem. This model will be illustrated on some particular classes of games.

Keywords : Game Theory, Constraint Satisfaction Problem (CSP), Nash Equilibrium, Best Response Strategies.