République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ESI ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
Ecole nationale Supérieure d' Informatique
ex. INI (Institut National de formation en Informatique)

Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État en Informatique

Option : Systèmes Informatiques

Thème

La théorie des jeux pour l'optimisation multi-objectif : application au problème d'affectation de fréquences

Réalisé par :
- BENGHERBIA Nawfel
- KHELIL Massyl Yacine

Encadré par :
- M^me^ BESSEDIK Malika

Soutenu le : 23 Juin 2016

devant le jury composé de :

- M. HADDADOU Hamid
- M^me^ TEMGLIT Nacera
- M^me^ AYAD Khadidja

Promotion : 2015/2016

Remerciements

Louanges à Dieu Tout Puissant.
Qu'Il nous pardonne et nous guide vers le droit chemin.

Tous nos remerciements vont à nos parents et nos familles envers lesquels nous serons toujours redevables. Ceux-ci nous ont apporté leur soutien, leurs encouragements et plus encore. Un grand merci.

Nous tenons à exprimer toute notre gratitude à M^{me} Malika Bessedik pour nous avoir encadrés avec une très grande disponibilité et de nous avoir éclairés de ses remarques et de ses précieux conseils.

Un grand merci aux membres de l'équipe optimisation du laboratoire LMCS et notamment M^{me} Fatima Benbouzid-Si Tayeb et à la directrice M^{me} Karima Benatchba pour leur accompagnement et leur bienveillance. Merci également à tous nos enseignants, à l'équipe pédagogique et à l'administration de l'ESI et particulièrement à M^{me} Dahbia Aït Ali Yahia, toujours présente quand nous avons besoin d'aide.

Nous somme très reconnaissants au CERIST et notamment à M. Ahcène Bendjoudi pour nous avoir permis d'utiliser le cluster IBNBADIS afin d'y effectuer les tests de notre système.

Nous tenons aussi à remercier publiquement les chercheurs qui ont la gentillesse de répondre à nos sollicitations, notamment pour des questions portant sur leurs publications, à savoir M. Carlos Segura, M. Timothy Ganesan, M. Francisco Luna et M. Eckart Zitzler.

Nos remerciement vont également à Mesdames et Messieurs les membres du jury qui nous font l'honneur d'examiner ce travail.

Comment ne pas remercier nos camarades pour leur aide, leurs contributions et leurs encouragements. Merci à M. Omar Kada, M. Imad Lamari, M. Aimene Bounab, M. Youcef Benachour, M. Abdelouahab Khelifati et M. Khalid Boufelfel.

Enfin, à tous ceux qui nous ont aidé : MERCI !

Résumé

La théorie des jeux, étudiant le comportement d'individus confrontés à des situations stratégiques, a vu ses techniques appliquées dans de nombreux domaines. Un exemple d'application est l'amélioration des méthodes d'optimisation multi-objectif. Nous nous proposons de considérer cet apport en l'employant sur un problème d'intérêt pratique très étudié qui est, en l'occurrence, le problème d'affectation de fréquences. Ce dernier, auquel sont confrontés les opérateurs téléphoniques, prend sa source de la dualité entre la demande en fréquences toujours grandissante des abonnés et le nombre de fréquences disponibles qui reste limité. La réutilisation des fréquences est nécessaire mais doit tenir compte des interférences générées. Après implémentation et tests de trois approches différentes basées sur la théorie des jeux, nous montrons que l'hybridation entre les notions d'équilibre de Nash et de Pareto-dominance permettent d'obtenir de meilleurs résultats que des algorithmes d'optimisation multi-objectifs tels *NSGA-II* et *SPEA2* qui constituent des références en la matière.

Abstract

Game theory, a field that studies the behaviour of individuals faced with strategic situations, saw it's techniques applied in many fields. An example of application of this theory is the improvement of multi-objective optimization methods. We suggest to consider this contribution by employing it on a very studied practical problem called the frequency assignment problem. It is faced by mobile network operators because of the duality between the frequency demands of subcribers, that is continuously growing, and the number of available frequency which remains limited. Therefore frequency reuse is necessary but must take into account the generated interference. After the implementation of three methods, we will demonstrate that the hybridization between the concepts of Nash equilibrium and Pareto-dominance can obtain better results than standard multi-objective optimization algorithms such as *NSGA-II* and *SPEA2*, two references in the field.

Table des matières

Remerciements i

Résumé ii

Abstract iii

Liste des figures viii

Liste des tableaux x

Liste des abréviations xi

Introduction générale 1

I Étude bibliographique 3

1 La théorie des jeux 4

Introduction 4

1.1 Terminologie 5

1.1.1 La théorie des jeux 5

1.1.2 Jeu et rationalité 5

1.1.3 Stratégie, stratégie dominée et stratégie dominante 5

1.1.4 Pareto-optimalité 6

1.2 Jeux sous forme normale (ou forme stratégique) 6

1.2.1 Définition 6

1.2.2 Équilibre de Nash en stratégie pure 7

1.2.3 Équilibre de Nash en stratégie mixte 9

1.2.4 Complexité de recherche des équilibres mixte et corrélé 11

1.3 Jeux sous forme étendue 12

1.3.1 Définition 12

1.3.2 Stratégies en forme étendue . . . 13
1.3.3 Transformation de la forme étendue vers forme normale . . . 13
1.3.4 Équilibre parfait en sous-jeu . . . 14
1.3.5 Jeux à information imparfaite . . . 15
1.3.6 Jeux à information incomplète . . . 16
1.4 Jeux répétés . . . 16
1.5 Jeux évolutionnaires . . . 17
1.5.1 La stabilité évolutionnaire . . . 17
1.5.2 La dynamique du réplicateur . . . 18
1.6 Jeux coopératifs . . . 19
1.6.1 La valeur de Shapley . . . 20
1.6.2 Le *Cœur* (*The Core*) . . . 21
1.7 Applications de la théorie des jeux . . . 22
Conclusion . . . 23

2 La théorie des jeux pour l'optimisation multi-objectif 24
Introduction . . . 24
2.1 Notions d'optimisation multi-objectif . . . 24
2.2 Comparaison des algorithmes d'optimisation multi-objectif . . . 26
2.3 Les algorithmes génétiques multi-objectifs . . . 28
2.3.1 Les algorithmes génétiques . . . 28
2.3.2 Exemples d'algorithmes génétiques pour l'optimisation multi-objectif 32
2.4 Travaux d'optimisation multi-objectif basés sur la théorie des jeux . . . 35
2.4.1 Travaux basés sur l'équilibre de Nash . . . 35
2.4.2 Travaux basés sur une hybridation Nash-Pareto . . . 37
2.4.3 Travaux basés sur les jeux coopératifs . . . 38
2.4.4 Travaux basés sur les jeux (co-)évolutionnaires . . . 39
2.4.5 Discussion . . . 39
Conclusion . . . 40

3 Le problème d'affectation de fréquences 41
Introduction . . . 41
3.1 Contexte . . . 42
3.2 Les réseaux de téléphonie mobile cellulaires . . . 43
3.2.1 Les interférences . . . 45
3.2.2 Contraintes imposées . . . 45
3.3 Définition du problème d'affectation de fréquences . . . 46
3.4 Modélisation par la coloration de graphe . . . 48
3.5 Variantes . . . 49
3.5.1 Affectation de spectre minimum *MS-FAP* (*Minimum Span FAP*) . . . 49

3.5.2 Affectation d'ordre minimum *MO-FAP* (*Minimum Order FAP*) . . 49
3.5.3 Affectation d'interférence minimum *MI-FAP* (*Minimum interference FAP*) 50
3.5.4 Affectation à spectre fixe (*Fixed Spectrum FAP*) 50
3.5.5 Affectation de perturbation minimale *PM-FAP* (*Perturbation-Minimizing FAP*) 50
3.5.6 Affectation de service maximum (*Max-FAP*) et Affectation de blocage minimum 50
3.6 Difficulté du problème 51
3.7 Affectations statique, dynamique et hybride 51
3.8 Méthodes de résolution existantes 52
3.8.1 Méthodes exactes 52
3.8.2 Heuristiques spécifiques 52
3.8.3 Méta-heuristiques 52
3.9 Les algorithmes génétiques pour le problème d'affectation de fréquences . . 53
3.9.1 La méthode de Valenzuela et al. (1998) 54
3.9.2 La méthode de Matsui et Tokoro (2000) 57
3.9.3 La méthode basée sur la représentation par clés aléatoires de Matsui et al. (2005) 59
3.9.4 La méthode de Matsui et al. (2003) pour le problème d'affectation de fréquence avec bande fixe 59
Conclusion 60

II Conception et réalisation 61

4 Conception 62
Introduction 62
4.1 Conception générale du système 62
4.2 Représentation de l'instance et de la solution du problème 63
4.3 Les fonctions objectifs 64
4.4 Notre adaptation de l'algorithme génétique 64
4.5 Approches de résolution adoptées 70
4.5.1 Approche basée sur l'équilibre de Nash (Sefrioui et Perlaux, 2000) . 70
4.5.2 Approche basée sur une hybridation Nash-Pareto (Lee et al., 2010) 73
4.5.3 Approche basée sur les jeux co-évolutionnaires (Sim et al., 2004) . . 75
Conclusion 77

5 Implémentation 78
Introduction 78

5.1 Langages de programmation et environnement d'exécution 78
5.2 Méthodologie de développpement 79
5.3 Réalisation 79
5.3.1 Lecture de l'instance Philadelphia 79
5.3.2 Lecture de l'instance COST 80
5.3.3 Décodage et opérations sur le chromosome : la classe Variator . . . 80
5.3.4 Evolution et visualisation des populations : l'interface GA 81
5.3.5 Programme principal 82
5.4 Parallélisation et optimisation du code 82
5.5 Interface graphique 83
Conclusion 83

III Tests et résultats **84**

6 Tests et résultats **85**
Introduction 85
6.1 Benchmarks de test utilisés 85
6.1.1 Philadelphia 86
6.1.2 COST 259 86
6.2 Choix des variateurs 86
6.3 Étude empirique 87
6.3.1 Méthodologie de l'étude empirique 87
6.3.2 Les algorithmes *NSGA-II* et *SPEA2* 88
6.3.3 Les algorithmes *Nash GA* et *Nash-Pareto GA* 91
6.3.4 L'algorithme *GCEA* 93
6.3.5 Discussion des résultats de l'étude empirique 94
6.4 Étude comparative 95
Conclusion 97

Conclusion générale **99**

Bibliographie **101**

Liste des figures

1.1 Exemple de jeu sous forme étendue 13
1.2 Exemple de deux sous-jeux (à l'intérieur des triangles) 15
1.3 Exemple de jeu sous forme étendue avec information imparfaite avec (en pointillés) un *ensemble d'informations* 15

2.1 Graphe illustrant le concept de la Pareto optimalité dans le cas de deux fonctions objectifs à minimiser. 26
2.2 Illustration de deux fronts de solutions 27
2.3 Graphe visualisant les niveaux de non-domination. 33

3.1 Exemple de modélisation hexagonale d'un réseau cellulaire 44
3.2 Exemple de réseau cellulaire à 3 cellules. La $1^{ère}$ cellule soumet 2 demandes de bandes de fréquences. Ces demandes sont notées A et B. La $2^{ème}$ cellule soumet 3 demandes (C, D et E) et la $3^{ème}$ soumet la demande F. Dans la matrice de séparation M donnée, $M_{i,j}$ représente la séparation minimale requise entre les fréquences affectées aux cellule i et j. 54
3.3 Décodage du chromosome $\{A, F, D, C, B, E\}$ (solution du problème de la figure 3.2) par l'algorithme de Valenzuela et al. (1998). 56
3.4 Interprétation du chromosome $\{1, 2, 1|2, 3, 2, 1|1, 2, 1|2, 1, 1\}$ en une permutation des demandes $\{A, B, C, D, E, F, G, H, I, J\}$. 58

4.1 Illustration du codage des solutions. 64
4.2 Partie 1 du décodage du chromosome $\{0.4, 0.7, 0.5, 0.2, 0.9, 0.3, 0.1, 0.25, 0.99, 0.6, 0.8, 0.33\}$ (solution du problème de la figure 3.2) par l'algorithme 15. 69
4.3 Partie 2 du décodage du chromosome $\{0.4, 0.7, 0.5, 0.2, 0.9, 0.3, 0.1, 0.25, 0.99, 0.6, 0.8, 0.33\}$ (solution du problème de la figure 3.2) par l'algorithme 15. 70

5.1 Exemple de fichier *log* généré donnant les paramètre de l'algorithmes ainsi que les valeurs des fonctions objectifs des solutions trouvées 82

6.1 Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *SPEA2* sur le benchmark *COST259* . 89
6.2 Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *SPEA2* sur le benchmark *Philadelphia* 90
6.3 Comparaison entre 3 combinaisons de paramètres faisant varier la probabilité de mutation de l'algorithme *NSGA-II* sur le benchmark *COST259* . 91
6.4 Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *Nash GA* sur le benchmark *COST259* . 92
6.5 Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *Nash Pareto GA* sur le benchmark *COST259* 93
6.6 Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de mutation et le nombre de générations de l'algorithme *GCEA* sur le benchmark *COST259* 94
6.7 Comparaison des algorithmes deux à deux avec 30 exécutions par algorithme 96

Liste des tableaux

1.1 Le jeu d'appariement (*matching pennies*) 8
1.2 La *bataille des sexes* 9
1.3 Le *dilemme du prisonnier* 9
1.4 Comparaison entre les niveaux d'équilibre du point de vue de l'existence et de la difficulté de recherche 12
1.5 La forme normale correspondante au jeu de la figure 1.1 14

2.1 Comparaison entre les deux fronts de solutions de la figure 2.2 27

5.1 Description de la structure *Philadelphia_instance* 79
5.2 Description de la structure *COST259* 80
5.3 Description de l'interface *Variator* 81
5.4 Description de l'interface *GA* 81

6.1 Valeurs des paramètres pour les algorithmes *NSGA-II* et *SPEA2* 88
6.2 Meilleurs paramètres pour les algorithmes *NSGA-II* et *SPEA2* 88
6.3 Valeurs des paramètres pour les algorithmes *Nash GA* et *Nash-Pareto GA* 91
6.4 Meilleurs paramètres pour les algorithmes *Nash GA* et *Nash-Pareto GA* . . 91
6.5 Valeurs des paramètres pour l'algorithme *GCEA* 93
6.6 Meilleurs paramètres pour l'algorithme *GCEA* 94
6.7 Comparaison de l'algorithme *Nash-Pareto GA* avec des travaux récents sur le benchmark *Philadelphia* 97

Liste des abréviations

BTS	***B**ase **T**ransceiver **S**ystem*
GSM	***G**lobal **S**ystem for **M**obile Communications*
LTE-A	***L**ong **T**erm **E**volution-**A**dvanced*
MSC	***M**obile **S**witching **C**enter*
PAF	**P**roblème d'**A**ffectation de **F**réquences
$\mathcal{PPAD}$	***P**olynomial **P**arity **A**rgument for **D**irected graphs*
UMTS	***U**niversal **M**obile **T**elecommunication **S**ystem*

Introduction générale

La théorie des jeux étudie le comportement d'individus confrontés à des situations stratégiques. Elle a été initiée et popularisée par Von Neumann et Morgenstern (1944) et Nash (1950) au milieu du siècle dernier et a vu ses outils mathématiques employés dans de divers domaines d'application allant de l'économie et de la politique à la modélisation des réseaux pair-à-pair et à l'amélioration des méthodes d'optimisation multi-objectif. C'est ce dernier point qui nous intéresse particulièrement. Pour nous y atteler, nous avons choisi le problème d'affectation de fréquences. Celui-ci se pose dans le domaine de la téléphonie sans fil, un domaine en plein essor et qui continue son expansion de part le monde.

La téléphonie mobile utilise des ondes électromagnétiques de fréquences précises afin de transmettre voix et données. Le nombre de fréquences utilisables est limité tandis que le nombre de demandes à satisfaire ne cesse d'augmenter. La réutilisation des fréquences est donc nécessaire. Or, les différents émetteurs du réseau doivent se faire affecter des fréquences sans que celles-ci ne puissent interférer entre-elles (ce qui se produit, par exemple, quand la même fréquence est allouée à deux émetteurs géographiquement proches). La réponse à cette conjugaison de contraintes définit le problème d'affectation de fréquences. C'est un problème d'optimisation combinatoire difficile, très étudié et possédant plusieurs métriques utilisables pour la mesure de la qualité des affectations (le nombre de demandes non satisfaites, le nombre de fréquences utilisées ou encore le taux d'interférence générée).

Les algorithmes génétiques figurent parmi les méthodes les plus utilisées en optimisation multi-objectif. Ainsi, nous avons implémenté trois méthodes de résolution basées sur la théorie des jeux sous forme d'algorithmes génétiques. Nous comptions, au départ n'utiliser que des représentations et des opérateurs simples afin de nous concentrer sur l'application de la théorie des jeux. Mais, chemin faisant, nous étions confrontés au fait que de nombreuses solutions violaient les contraintes du problème ; nous avons donc opté pour une approche avec décodeur permettant de ne générer que des solutions respectant l'ensemble des contraintes du problème. Dans la perspective de pouvoir comparer notre travail avec les algorithmes génétiques multi-objectifs standards sur la même configuration matérielle, nous avons implémenté deux des plus performants d'entre eux, à savoir les algorithmes *NSGA-II* et *SPEA2*. De plus, afin de pouvoir mieux valider notre travail

et le comparer à un maximum de travaux, nous avons décidé d'utiliser deux benchmarks (radicalement différents) du problème d'affectation de fréquences appelés *Philadelphia* et *COST259* comme base de comparaison.

Nous nous proposons d'utiliser des mécanismes issus de la théorie des jeux pour la résolution multi-objectif du problème d'affectation de fréquence (notons que ce problème s'applique aussi bien aux réseaux de deuxième que de troisième génération) ; le but étant de mesurer l'apport de la théorie des jeux sur un problème pratique. Nous verrons que la complémentarité entre la notion centrale de la théorie des jeux : l'équilibre de Nash, et la notion de Pareto-optimalité permettent d'améliorer les résultats des algorithmes d'optimisation multi-objectif.

Ce mémoire est composé de trois parties. La première consiste en étude bibliographique des différents aspects de notre travail en 3 chapitres. La théorie des jeux fait l'objet du premier d'entre eux. Nous y mettons l'accent sur les types de jeu qui nous intéresseront particulièrement dans la suite (à savoir les jeux sous forme normale et les jeux évolutionnaires). La chapitre 2 est consacré à l'utilisation de la théorie des jeux pour l'optimisation multi-objectif. Nous commençons par donner quelques notions d'optimisation multi-objectif ainsi que des exemples d'algorithmes (et particulièrement ceux basés sur des algorithmes génétiques) puis nous classifions les différents travaux selon le type de jeu utilisé. Le chapitre 3 aborde le problème d'affectation de fréquences avec ses différents objectifs avec un traitement particulier de l'utilisation des algorithmes génétiques pour la résolution du problème. Dans la partie 2, nous abordons la conception de notre système et des approches de résolution que nous avons choisies qui sont les algorithmes *Nash GA*, *GCEA* et *Nash-Pareto GA* (chapitre 4), puis la manière avec laquelle nous avons implémenté ces méthodes (chapitre 5). Enfin la dernière partie est consacrée aux tests et aux résultats que nous avons obtenus sur les benchmarks choisis (chapitre 6). Résultats que nous commentons et discutons avant de conclure.

Première partie

Étude bibliographique

Chapitre 1

La théorie des jeux

Introduction

La théorie des jeux propose un ensemble de mécanismes permettant d'étudier l'interaction d'agents humains (entreprises, institutions, gouvernements, traders,...) ou de machines poursuivant leurs objectifs propres. C'est après la publication du livre *Theory of Games and Economic Behavior* de Von Neumann et Morgenstern (1944) qu'elle a connu un développement de plus en plus large. Les résultats probants qu'elle a apportés en économie depuis le milieu du siècle dernier (pas moins de dix prix Nobel d'économie ont récompensé des travaux de théorie des jeux) ne sont pas arrêtés à ce seul domaine, mais ont porté également vers des applications aussi variées que la modélisation des réseaux ou l'optimisation combinatoire.

Ce chapitre vise à exposer la terminologie, les concepts de la théorie des jeux et les notions de base de ce domaine (section 1.1). Les différentes composantes du jeu (joueurs, actions et utilités) y sont également définies. Nous aborderons aussi différents types de jeux et leurs caractéristiques. Ainsi, si les actions sont prises simultanément (comme pour le jeu *pierre-papier-ciseaux* [1]), on parle de jeu sous forme normale (section 1.2). Si, par contre, les joueurs jouent tour à tour, on parle de jeux sous forme étendue (section 1.3). Dans ces deux types de jeu, une hypothèse de travail est introduite : les joueurs n'ont pas la possibilité de coopérer. Dans le cas contraire, les regroupements de joueurs sont modélisés par les jeux coalitionnels (section 1.6), par opposition aux jeux non-coopératifs.

1. Deux joueurs doivent choisir, simultanément, un de ces trois objets. Il y a match nul s'ils choisissent la même chose. Autrement, la *pierre* bat les *ciseaux* qui battent le *papier* qui, lui, bat la *pierre*

1.1 Terminologie

1.1.1 La théorie des jeux

La théorie des jeux est un outil mathématique modélisant le comportement d'agents **rationnels**, appelés **joueurs**, dans des situations d'interaction stratégique. C'est à dire lorsque le gain de chaque joueur dépend, non seulement, de ses propres actions, mais aussi de celles des autres. Cette théorie offre des outils d'analyse aussi bien aux concepteurs des jeux qu'aux joueurs (Başar, 2015).

1.1.2 Jeu et rationalité

Un **jeu** est décrit par l'ensemble de ses règles : le nombre de joueurs, les **actions** possibles par chaque joueur et le profit (ou gain) des joueurs à chaque issue (ou résultat) du jeu.

Pour éviter toute confusion que pourrait créer le fait qu'un gain, un paiement ou un profit puisse être négatif ou nul, on parle plus souvent d'**utilité** que de gain ou de profit (l'utilité qu'a un joueur d'effectuer une action ou la fonction *utilité* associée à une action). Conventionnellement, les joueurs cherchent à maximiser leur fonction d'utilité (nous donnerons des exemples complets en 1.2).

La théorie des jeux suppose que les joueurs se comportent **rationnellement** : c'est à dire qu'ils agissent au mieux de leurs intérêts dans le cadre des informations qu'ils possèdent (Bouyaux, 2014). Pour ce faire, selon Osborne et Rubinstein (1994), une hypothèse de travail consiste à dire qu'un joueur rationnel connaît les actions permises, possède des préférences claires concernant les issues du jeu, forme des attentes au sujet des possibles inconnues et choisit son action délibérément, après un certain processus d'optimisation.

1.1.3 Stratégie, stratégie dominée et stratégie dominante

Une **stratégie** est un plan d'action complet pour toute situation pouvant se produire durant le jeu. On dit qu'une stratégie est **pure** lorsque la probabilité de choix de toutes les actions est soit 0, soit 1. Sinon, on dit que la stratégie est **mixte** (Bouyaux, 2014). Ainsi, dans le jeu *pierre-papier-ciseaux*, chaque joueur possède trois stratégies pures (une par action) et une infinité de stratégies mixtes (par exemple choisir chacune des actions avec une probabilité égale à $\frac{1}{3}$).

On dit qu'une stratégie s est **dominée** par une autre stratégie s' du même joueur si le profit généré par la stratégie s' est meilleur que celui généré par s quelles que soient les stratégies des autres joueurs. Une stratégie qui domine toute autre stratégie du même joueur est dite stratégie **dominante**. Si elle est meilleure ou égale aux autres, on parle alors de stratégie **faiblement dominante** et si elle est strictement meilleure on parle de

stratégie **strictement dominante**.

Les stratégies dominantes sont puissantes à la fois d'un point de vue analytique que pratique. Ainsi, le joueur n'a pas besoin de prédire ce que feront les autres joueurs, il a une meilleure stratégie bien définie (Jackson, 2011).

1.1.4 Pareto-optimalité

Lorsqu'on s'intéresse à analyser les jeux, du point de vue d'un observateur neutre, il est naturel de chercher à maximiser les gains de tous les participants. Ceci est similaire au concept de **Pareto-optimalité** introduit par l'économiste Italien Vilfredo Pareto en 1869 dans le domaine de l'économie du bien-être (*welfare economics*).

Dans un système contenant une variable de décision (ou variable d'entrée) vectorielle $x \in X \subseteq \mathbb{R}^n$ et m critères de performance (ψ_j) tels que $\psi_j : \mathbb{R}^n \to \mathbb{R},\ j = 1, ..., m$, la décision $x^* \in X$ est **Pareto-optimale** s'il est impossible d'améliorer un critère j, sans en détériorer un autre. C'est à dire $\psi_j(x) \geq \psi_j(x^*)\ \forall j = 1, ..., m \Longrightarrow \psi_j(x) = \psi_j(x^*)\ \forall j = 1, ..., m$

Ce concept d'optimisation vectorielle (optimiser un vecteur de fonctions) peut être exploité pour analyser les jeux du point de vue d'un observateur neutre ou des joueurs en coopération (Haurie, 2015).

S'il existe une situation qui engendre un gain supérieur ou égal à celui d'une autre pour tous les joueurs, on dit que la première situation **Pareto-domine** la seconde. Cette seconde situation est dite **Pareto-dominée**. Elle n'est pas préférable du fait qu'elle détériore le profit d'au moins un des joueurs. Une situation *Pareto-optimale* n'est *Pareto-dominée* par aucune autre. Il ne faut pas confondre la dominance (au sens de stratégie dominée et dominante) et la *Pareto-dominance* qui sont deux concepts distincts. Cette différence est parfaitement illustrée par le jeu appelé *le dilemme du prisonnier* qui sera défini en détail au niveau de la sous-section 1.2.2.

1.2 Jeux sous forme normale (ou forme stratégique)

1.2.1 Définition

Un **jeu sous forme normale** modélise la situation dans laquelle les joueurs choisissent, définitivement, un plan d'action puis effectuent une seule action simultanément [2]. Formellement, selon Osborne et Rubinstein (1994), un jeu sous forme normale consiste en :

2. La simultanéité n'est pas un impératif en soi. Cela exprime plus le fait que les joueurs ne sont pas informés des actions des autres et qu'ils prennent leur décision de manière indépendante.

— Un ensemble (fini) de **joueurs** noté par un ensemble d'entiers $N = \{1, ..., n\}$; avec $n \geq 2$;

— Pour chaque joueur $i \in N$ un ensemble non vide A_i d'**actions** possibles pour ce joueur. Un **profil d'actions** $a = (a_1, ..., a_n) \in A$, avec A le produit cartésien des A_i. On note par commodité $a = (a_i, a_{-i})$ où $a_{-i} = (a_1, ..., a_{i-1}, a_{i+1}, ..., a_n)$. C'est à dire le couple (action du joueur i, actions des autres joueurs). Dans l'exemple du tableau 1.1 les joueurs ont deux actions possibles *Pile* ou *Face* ;

— Une **fonction d'utilité** $u_j(a_i, a_{-i})$ à valeurs dans l'ensemble des réels. Elle traduit l'intérêt porté par le joueur j à l'action a_i quand les autres joueurs choisissent les actions a_{-i} (Bouyaux, 2014). L'interaction entre les joueurs est ici bien visible. Toujours dans le même exemple, on a, $u_1(Pile, Pile) = -1$ et $u_2(Pile, Pile) = 1$.

L'action de chaque joueur répond en fait à sa propre **stratégie**, notée s_i. On obtient ainsi un **profil de stratégies** $s = (s_1, ..., s_n) \in S$, avec S le produit cartésien des S_i. On note également par commodité $s = (s_i, s_{-i})$ où $s_{-i} = (s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$. La notion de **stratégie pure** est (dans le cas de jeux sous forme normale) équivalente à la notion d'action puisqu'il s'agit de faire un choix unique entre l'ensemble des actions possibles pour un joueur. La différence entre stratégie et action se fera plus nettement sentir lorsque nous aborderons les stratégies mixtes (en 1.2.3).

Un jeu peut être représenté sous forme normale, s'il n'y a pas de liens stratégiques entre les joueurs. C'est à dire qu'on ne considère que les fonctions d'utilité et non les effets d'une action sur le comportement futur des autres joueurs (Osborne et Rubinstein, 1994).

Si tous les A_i sont finis, le jeu est dit fini (Bouyaux, 2014). Un jeu fini sous forme normale avec deux joueurs est représenté par une bi-matrice (comme le tableau 1.1). Le premier joueur est le joueur de ligne, le second le joueur de colonne. Chaque cellule du tableau représente une issue possible du jeu. Pour une issue donnée, on inscrit les utilités des joueurs ainsi : (utilité du joueur 1, utilité du joueur 2).

L'exemple du tableau 1.1 représente le *jeu d'appariement* où les joueurs doivent choisir entre deux alternatives (*Pile* ou *Face*). Le joueur 2 souhaite que les actions soient différentes entre-elles tandis que le joueur 1 veut qu'elles soient les mêmes.

On dit qu'un jeu à deux joueurs sous forme normale est **à somme nulle** si la somme des utilités des joueurs est nulle pour toutes les issues (c'est le cas pour le *jeu d'appariement*). Les jeux à somme nulle sont dits **purement compétitifs** car les intérêts des joueurs sont diamétralement opposés (Ozdaglar, 2015).

1.2.2 Équilibre de Nash en stratégie pure

Une fois le jeu modélisé, la théorie des jeux cherche à prédire le comportement des joueurs. Sachant qu'ils sont rationnels, les joueurs vont employer l'ensemble des informations à leur disposition en essayant de prédire les actions des autres joueurs (également

Joueur1\Joueur2	Pile	Face
Pile	(-1, 1)	(1, -1)
Face	(1, -1)	(-1, 1)

TABLEAU 1.1 – Le jeu d'appariement (*matching pennies*)

rationnels). Un joueur donné va ainsi suivre le raisonnement suivant qui est une forme de récursion infinie : «Je prédis ce que les autres vont faire sachant que chacun d'entre eux va prédire ce que les autres (moi compris) allons faire, sachant que chacun ... etc.»

Cette récursion admet comme solution (on parle de concept de solution) une situation appelée **équilibre de Nash en stratégie pure** où chaque joueur sélectionne sa meilleure stratégie compte tenu des stratégies des autres joueurs (Bouyaux, 2014).

Plus rigoureusement, un profil de stratégie pure $s = (s_i, s_{-i})$ est un équilibre de Nash en stratégie pure si pour tous les joueurs i, la stratégie s_i est une meilleure réponse à s_{-i} c'est à dire que $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$ pour tout s'_i (Ozdaglar, 2015).

La définition montre bien qu'un profil de stratégie dominant (constitué de stratégies dominantes) est un équilibre de Nash en stratégie pure (l'inverse n'est pas forcément vrai) (Jackson, 2011).

Une interprétation de l'équilibre consiste à dire que chaque joueur agit en conséquence de sa prévision des actions (Osborne et Rubinstein, 1994) de sorte qu'aucun d'entre eux n'a intérêt à modifier unilatéralement son comportement ; d'où la dénomination d'*équilibre*. Soulignons que les joueurs sont supposés ne pas pouvoir s'entendre sur la marche à suivre (nous discuterons des jeux où la coopération est admise au niveau de la section 1.6).

Tous les jeux sous forme normale ne possèdent pas forcément d'équilibre de Nash en stratégie pure. Ainsi, dans l'exemple du jeu d'appariement (tableau 1.1), pour chaque issue, le joueur se voyant attribuer l'utilité -1, a intérêt à changer de stratégie. Dans la nouvelle issue, c'est l'autre joueur qui a intérêt à changer et ainsi de suite sans pouvoir atteindre un quelconque équilibre.

Certains jeux possèdent plusieurs équilibres de Nash en stratégie pure, c'est le cas du jeu *bataille des sexes* présenté dans le tableau 1.2. Les joueurs (un homme et son épouse) souhaitent regarder un film. L'homme préfère le film 1 et la femme le film 2 (L'homme préfère l'issue (film 1, film 1) où il obtient l'utilité 2 à l'issue (film 2, film 2) où il obtient 1, et inversement pour la femme), mais tous deux veulent, par dessus tout, être ensemble et ainsi avoir des actions coordonnées (leur utilité est nulle pour les issues (film 1, film 2) et (film 2, film 1)). Les cas ou les deux font la même activité sont tous deux des équilibres de Nash en stratégie pure. Dans les cas où plusieurs équilibres existent, comme ici, la prédictibilité du jeu est amoindrie et la notion même d'équilibre perd de son sens.

De plus, la nature même de la définition de l'équilibre ne garantit en aucun cas que l'issue du jeu soit Pareto-dominante, et ce, en dépit du fait que tous les joueurs agissent par

Homme \ Femme	Film 1	Film 2
Film 1	(2, 1)	(0, 0)
Film 2	(0, 0)	(1, 2)

TABLEAU 1.2 – La *bataille des sexes*

Joueur 1\Joueur 2	Ne pas dénoncer	Dénoncer
Ne pas dénoncer	(-1, -1)	(-3, 0)
Dénoncer	(0, -3)	(-2, -2)

TABLEAU 1.3 – Le *dilemme du prisonnier*

une meilleure réponse. Ce fait est bien illustré par le *dilemme du prisonnier* du tableau 1.3.

La situation de ce dilemme est la suivante : la police arrête deux suspects, mais n'a pas assez de preuves pour les inculper. Ils sont séparés et se voient proposer un marché. Si un seul dénonce, il est libéré tandis que l'autre prend une lourde peine. Si les deux dénoncent, ils prennent une peine moyenne. Si aucun ne dénonce l'autre, les deux prennent une peine légère. Une infinité de fonctions d'utilité peut modéliser ce jeu. Un exemple est donné dans le tableau 1.3.

Toutes les issues du jeu sauf l'équilibre de Nash en stratégie pure (Dénoncer, Dénoncer) sont Pareto-optimales. De plus, la différence entre les utilités de l'équilibre et les autres peut être rendue arbitrairement grande ; cet un exemple typique où l'équilibre n'offre aucune garantie d'efficacité au sens de Pareto (Ozdaglar, 2015).

1.2.3 Équilibre de Nash en stratégie mixte

Pour les jeux ne possédant pas d'équilibre de Nash en stratégie pure, une autre forme d'équilibre, moins prédictive mais plus générale, peut être définie. En essayant de comprendre comment il serait possible de bien jouer au *jeu d'appariement*, on peut se convaincre que pour espérer gagner, il faudrait que l'adversaire ne puisse pas deviner l'action qui sera choisie. Il s'agit donc d'introduire de l'aléa dans le choix des actions. On parle de stratégie mixte.

Une **stratégie mixte** pour un joueur i est une distribution de probabilité s_i sur ses actions a_i. Ainsi, $s_i(a_i)$ désigne la probabilité que l'action a_i soit choisie. La notion d'équilibre de Nash reste la même avec cette redéfinition de la notion de stratégie. On parle ici d'**équilibre de Nash en stratégie mixte** ou d'**équilibre mixte**. La différence réside au niveau de la fonction utilité qui devient une utilité moyenne $u_i(s) = \sum_{a \in A} u_i(a) P(a/s)$ avec $P(a/s) = \prod_{j \in N} s_j(a_j)$ les distributions de probabilités utilisées par les joueurs (qui sont indépendantes) (Leyton-Brown et Shoham, 2008).

La définition de l'équilibre mixte impose que, pour chaque joueur le choix des actions effectuées par ses adversaires l'indiffère. Si ce n'était pas le cas, il pourrait choisir, avec une plus grande probabilité, la réponse qui lui rapporterait le plus (cette déviation contredirait la définition de l'équilibre).

Le résultat central, que l'on doit à Nash (1950), est que **tout jeu fini possède un équilibre mixte**. Nash a étendu le résultat, qui était déjà connu pour 2 joueurs (et notamment par Von Neumann et Morgenstern (1944)) à un nombre quelconque de joueurs.

La notion de stratégie mixte englobe celle de stratégie pure (en prenant une probabilité égale à 1 pour l'action considérée) (Jackson, 2011). L'équilibre mixte n'arrive, certes, pas à prédire l'action des joueurs mais il décrit la régularité des probabilités du choix des actions.

L'existence d'un ou de plusieurs équilibres de Nash en stratégie pure ne veut pas dire que des équilibres mixtes non-purs n'existent pas. À titre d'exemple, la *bataille des sexes* possède un tel équilibre mixte (avec $s_1(film\ 1) = s_2(film\ 2) = \frac{2}{3}$ et $s_1(film\ 2) = s_2(film\ 1) = \frac{1}{3}$). On peut vérifier qu'un joueur est indifférent entre les alternatives A ou B car il obtient une utilité de $\frac{2}{3}$ en choisissant l'une ou l'autre des deux actions, étant donné l'aléa introduit par son adversaire.

A Équilibre corrélé

En observant l'équilibre mixte trouvé dans la *bataille des sexes*, on remarque qu'il serait judicieux d'éviter qu'à l'issue du jeu, les deux joueurs aient effectué des actions différentes. Dans la pratique, un couple pourrait s'entendre de sorte à choisir aléatoirement l'une ou l'autre des solutions consistant à avoir des actions identiques ($(film\ 1, film\ 1)$ ou $(film\ 2, film\ 2)$) avec une probabilité égale à $\frac{1}{2}$. Cette situation est modélisée en corrélant les distributions aléatoires des actions. Ceci généralise la notion d'équilibre de Nash mixte en supprimant la condition d'indépendance des distributions.

D'un point de vue externe au jeu, c'est comme si l'issue était choisie aléatoirement et d'une manière équitable (de sorte que cela soit un équilibre) en informant les joueurs de celle-ci. Ces derniers ne sont pas forcés de jouer comme il leur est indiqué, mais n'ont aucun intérêt à faire autrement.

L'intérêt de l'équilibre corrélé est d'assurer une meilleure utilité moyenne aux joueurs par rapport à l'équilibre mixte. Pour l'exemple de la *bataille des sexes* l'équilibre mixte rapporte, à chaque joueur, une utilité moyenne de $\frac{1}{3} \times \frac{2}{3} \times 2 + 0 + 0 + \frac{2}{3} \times \frac{1}{3} \times 1 = \frac{2}{3}$ tandis que l'équilibre corrélé $\frac{1}{2} \times 2 + \frac{1}{2} \times 1 = \frac{3}{2}$.

1.2.4 Complexité de recherche des équilibres mixte et corrélé

La littérature étudie l'appartenance des problèmes aux différentes classes de complexité, dont les plus utilisées sont les classes $\mathcal{P}$ et $\mathcal{NP}$. La classe $\mathcal{P}$ englobe les problèmes solvables en temps polynomial [3]. La classe NP contient les problèmes pour lesquels un algorithme polynomial de validation de solution existe. Il est clair que $\mathcal{P} \subseteq \mathcal{NP}$ mais une des grandes questions ouvertes est de savoir si $\mathcal{P} = \mathcal{NP}$. Les théoriciens de la complexité algorithmique pensent en majorité que $\mathcal{P} \neq \mathcal{NP}$ [4]. Parmi les problèmes dans $\mathcal{NP}$, Les problèmes dits $\mathcal{NP}-$difficiles sont ceux qui opposent le plus de difficultés. Résoudre l'un d'entre eux en temps polynomial permettrait de résoudre l'ensemble des problèmes de $\mathcal{NP}$ en temps polynomial.

L'équilibre en stratégie pure n'existant pas toujours, la littérature s'est plus intéressée au problème de recherche de l'équilibre de Nash en stratégie mixte, appelé problème *MNE* (pour *Mixed Nash Equilibrium*).

Megiddo et Papadimitriou (1991) ont montré que si le problème *MNE* est $\mathcal{NP}-$complet alors la conjecture $\mathcal{NP} \neq co\mathcal{NP}$ [5] serait fausse (un résultat qui serait étonnant pour la plupart des chercheurs). Cela a réorienté les travaux vers la recherche d'une classe incluse dans $\mathcal{NP}$ et où *MNE* serait complet (c'est à dire aussi difficile que tout autre problème de la même classe) (Roughgarden, 2013).

La classe en question a été trouvée et s'appelle $\mathcal{PPAD}$ [6]. Elle formalise l'ensemble des problèmes pouvant être réduits au problème dit *End of the line*, formulé comme suit : "Étant donné un graphe orienté (de grandeur exponentielle en terme de la taille de l'instance du problème à réduire) avec un sommet de degré entrant différent du degré sortant. On sait qu'il existe un autre sommet ayant la même propriété." Cet argument est appelé *argument de parité*. La recherche de cet autre sommet, qui est au pire exponentielle, définit la classe $\mathcal{PPAD}$ (Daskalakis et al., 2009). On utilise en général des algorithmes tels celui de *Lemke–Howson* (Lemke et Howson, 1964) pour la résolution de ce type de problème.

La recherche de l'équilibre de Nash est $\mathcal{PPAD}-$difficile (Papadimitriou, 1994) (c'est à dire qu'elle est aussi difficile que tous les autres problèmes $\mathcal{PPAD}$). La plupart des autres problèmes connus pour être $\mathcal{PPAD}-$difficiles sont d'un ordre théorique assez avancé. Citons le lemme de Sperner et les problèmes de recherche de point fixe de Brouwer, de Kakutani et de Borsuk-Ulam (le lecteur intéressé peut se référer à (Papadimitriou, 1994; Daskalakis et al., 2009)).

La difficulté de recherche de l'équilibre de Nash diminue de son pouvoir prédictif car les joueurs (machines ou humains) ne peuvent pas le déterminer efficacement. La littérature

3. C'est à dire d'une manière efficace même si la taille des instances augmente.
4. www.cs.umd.edu/~gasarch/papers/poll.pdf
5. $co\mathcal{NP}$ est la classe des problèmes pour lesquels on peut vérifier qu'une donnée n'est pas une solution au problème en temps polynomial
6. pour *Polynomial Parity Argument for Directed graphs*

Équilibre	**Existence**	**Difficulté calculatoire**
Équilibre de Nash en stratégie pure	N'existe pas toujours	Difficile
Équilibre de Nash en stratégie mixte	Existe toujours	Difficile ($\mathcal{PPAD}$-complet)
Équilibre corrélé	Existe toujours	Complexité polynomiale

TABLEAU 1.4 – Comparaison entre les niveaux d'équilibre du point de vue de l'existence et de la difficulté de recherche

suggère d'aller (quand le contexte le permet) vers d'autres types de solutions, comme par exemple l'équilibre corrélé. Plusieurs résultats concernant cet équilibre se trouvent être polynomiaux quand ceux de l'équilibre de Nash sont $\mathcal{NP}-$difficiles (Roughgarden, 2009). Le tableau 1.4 résume les degrés de difficulté des trois types d'équilibre précités.

1.3 Jeux sous forme étendue

1.3.1 Définition

Contrairement à la forme normale qui ne prend pas le temps et le séquencement des actions en considération, la forme étendue (ou développée) rend la structure temporelle du jeu explicite et permet de représenter les jeux où les participants jouent à tour de rôle.

Deux variantes principales des jeux sous forme étendue existent : jeux à information parfaite, et jeux à information imparfaite. C'est la première variante que nous nous proposons d'explorer dans cette section. La variante à information imparfaite fait l'objet de la section 1.3.5.

Formellement, selon Leyton-Brown et Shoham (2008), un **jeu sous forme étendue** à information parfaite peut être représenté par le tuple $(N, A, H, Z, \chi, \rho, \sigma, u)$ formé de :

- Un ensemble (fini) de **joueurs** $N = \{1, ..., n\}$;
- Un ensemble non vide A d'**actions** possibles pour tous les joueurs ;
- Pour représenter la structure temporelle du jeu, l'**arbre de jeu** est défini. Les joueurs prennent leur tour, un par un, jusqu'à arriver à un nœud où le jeu se termine. L'arbre est modélisé par :
  - Des nœuds de décisions (ou de choix) H où un des joueurs doit choisir une action : ce sont les nœuds internes de l'arbre ;
  - La fonction $\chi : H \to A$ qui affecte à chaque nœud de choix un ensemble d'actions possibles ;
  - La fonction $\rho : H \to N$ qui affecte à chaque nœud de choix le joueur dont c'est le tour ;

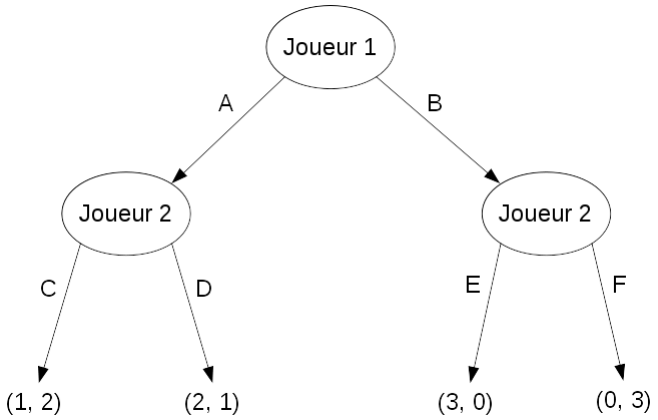


FIGURE 1.1 – Exemple de jeu sous forme étendue

— Des nœuds terminaux Z (les feuilles de l'arbre) qui sont les issues du jeu. Ils sont étiquetés avec les utilités des joueurs ;

— La fonction successeur $\sigma : H \times A \to H \cup Z$ qui fait correspondre à chaque paire (nœud de choix, action) un nœud destination (cette fonction représente les branches de l'arbre). Le nouveau nœud peut être terminal ou de choix, ainsi ($\forall h_1, h_2 \in H$ et $\forall a_1, a_2 \in A$, si $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ alors $h_1 = h_2 \wedge a_1 = a_2$) ;

— La **fonction d'utilité** $u = (u_1, ..., u_n); u : Z \to \mathbb{R}^{|N|}$ qui donne l'utilité de tout nœud terminal dans Z.

À titre d'exemple, la figure 1.1 présente l'arbre correspondant à un jeu sous forme étendue où le joueur 1 commence et a le choix entre les actions A et B. Le joueur 2 peut, ensuite, choisir entre les actions C et D si le joueur 1 a joué A, ou bien, entre les actions E et F dans le cas contraire. Les nœuds feuilles contiennent les vecteurs représentant les utilités des joueurs.

1.3.2 Stratégies en forme étendue

Dans un jeu sous forme étendue, une stratégie pure pour un joueur est un profil d'actions sur chacun de ses nœuds de décision. Informellement, c'est l'ensemble des informations qu'il devrait communiquer à un tiers afin que celui-ci joue exactement comme lui.

Étant donnée cette définition d'une stratégie pure, on peut réutiliser la définition de la stratégie mixte en forme normale : Une stratégie mixte est une distribution de probabilité sur les stratégies pures possibles (Leyton-Brown et Shoham, 2008).

1.3.3 Transformation de la forme étendue vers forme normale

Il est naturel de se demander si on peut transformer les jeux de la forme étendue vers la forme normale afin d'en exploiter les résultats. Cette transformation existe ; cependant,

J 1 \ J 2	CE	CF	DE	DF
A	(1, 2)	(1, 2)	(2, 1)	(2, 1)
B	(3, 0)	(0, 3)	(3, 0)	(0, 3)

TABLEAU 1.5 – La forme normale correspondante au jeu de la figure 1.1

elle n'est pas réversible car le séquencement des actions est perdu dans la forme normale.

Pour illustrer cette transformation, la forme normale associée au jeu de la figure 1.1 est donnée par le tableau 1.5. Les stratégies pures du joueur 1 sont A et B et celles du joueur 2 sont CE, CF, DE et DF. La notation CE signifie que le joueur 2 jouera C si le joueur 1 joue A et E dans l'autre cas.

Il résulte de l'existence de cette transformation que l'équilibre de Nash en stratégie mixte pour un jeu sous forme étendue existe toujours. En fait, il y a un théorème plus fort qui énonce que tout jeu sous forme étendue à information parfaite admet un équilibre de Nash en stratégie pure (Leyton-Brown et Shoham, 2008).

Il est possible d'exploiter cette transformation pour calculer l'équilibre de Nash des jeux sous forme étendue. Cependant, cet approche n'est pas utilisée en pratique car le nombre de stratégies pures croît exponentiellement avec la taille de l'arbre (Castañón, 2015). De plus, l'équilibre de Nash n'est pas le concept de solution approprié dans ce type de jeux, il est délaissé au profit de l'équilibre parfait en sous-jeu qui en est un cas particulier.

1.3.4 Équilibre parfait en sous-jeu

Pour un jeu G sous forme étendue, on définit un **sous-jeu** ayant pour racine le nœud h comme étant la restriction du jeu G aux descendants de h.

Un équilibre de Nash du jeu sous forme étendue à information parfaite G est dit **parfait en sous-jeu** si sa restriction à tout sous-jeu G' de G reste un équilibre de Nash. Un tel équilibre en stratégie pure existe toujours (Castañón, 2015).

Dans le jeu représenté par la figure 1.2, le profil de stratégie (A, DF) est un équilibre de Nash, car aucun joueur n'a intérêt à changer sa stratégie compte tenu de la stratégie de l'autre. Cependant, cet équilibre n'est pas parfait en sous-jeu car le choix de l'action F par le joueur 2 n'est pas *crédible* : Le joueur 2 n'a pas intérêt à jouer F dans le sous-jeu 2. Le profil de stratégie (B, DE) par contre est un équilibre de Nash parfait en sous-jeu.

Pour trouver un équilibre de Nash parfait en sous-jeu, on utilise l'algorithme du **raisonnement rétrograde** (ou *backward induction*) (Von Neumann et Morgenstern, 1944). Cet algorithme, semblable à une approche par programmation dynamique et de complexité polynomiale (Szymanik, 2013), associe, à chaque nœud de l'arbre du jeu, un vecteur d'utilité. L'algorithme remonte à partir des feuilles (qui ont une utilité connue au départ), en

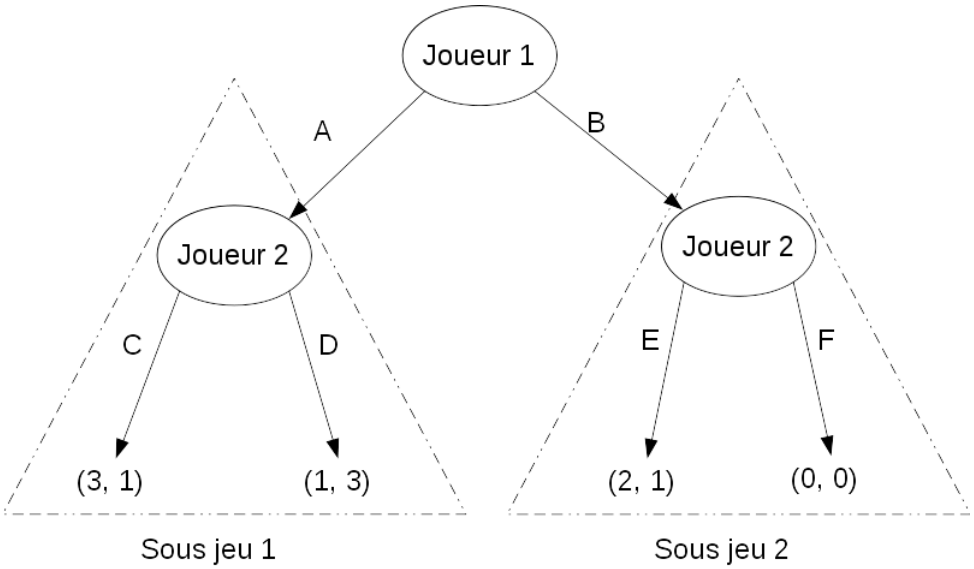


FIGURE 1.2 – Exemple de deux sous-jeux (à l'intérieur des triangles)

étiquetant les nœuds jusqu'à la racine. Ensuite, pour construire un équilibre de Nash parfait en sous-jeu, on simule le jeu de sorte que chaque joueur choisisse l'action maximisant son utilité (qu'il lit dans les nœuds intermédiaires) à chaque étape du jeu.

1.3.5 Jeux à information imparfaite

Dans les jeux sous forme étendue à information imparfaite, les joueurs peuvent ne pas avoir connaissance de l'ensemble des actions précédemment effectuées (par les autres joueurs ou par eux mêmes). Ce modèle, plus riche en possibilités, permet d'englober les situations où un joueur oublie l'action d'un de ces adversaires ou n'a, tout simplement, pas accès à cette information. Les jeux impliquant des machines en sont un exemple d'application concret. La gestion mémoire peut, en effet, contraindre à ne pas sauvegarder l'ensemble des actions précédentes (Osborne et Rubinstein, 1994).

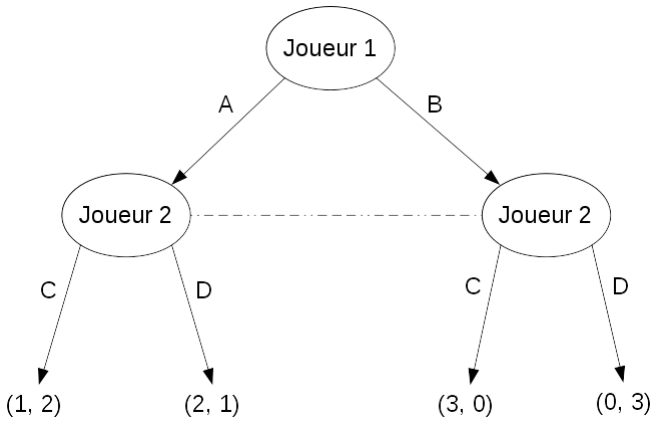


FIGURE 1.3 – Exemple de jeu sous forme étendue avec information imparfaite avec (en pointillés) un *ensemble d'informations*

On représente l'appartenance à un même ensemble d'informations par des lignes pointillées entre les nœuds, comme illustré dans la figure 1.3. Les deux nœuds de choix du joueur 2 sont dans le même ensemble d'informations et offrent la même possibilité d'actions. Ce qui veut dire que le joueur 2 ne sait pas si le joueur 1 a choisi A ou B au moment de faire son choix entre C et D.

1.3.6 Jeux à information incomplète

Jusqu'à présent, nous avons supposé que les données du jeu (et notamment celles concernant la fonction d'utilité) étaient connues de tous. Or, dans la pratique, ce n'est pas toujours le cas. Un jeu est dit **Bayésien** (ou à **information incomplète**) quand un joueur ne connaît pas forcément l'utilité des autres joueurs (Leyton-Brown et Shoham, 2008). L'exemple typique est celui de la vente aux enchères. Un acheteur potentiel (notre joueur) ne sait pas à combien les autres évaluent les objets proposés à la vente et ne connaît donc pas leur fonction d'utilité.

1.4 Jeux répétés

Les jeux répétés analysent, sur le long terme, les interactions existantes entre les joueurs en prenant en compte des phénomènes multiples (comme la vengeance ou les menaces). Ils réitèrent à plusieurs reprises un même jeu sous forme normale dit *jeu constituant* (Osborne et Rubinstein, 1994), comme, par exemple, le jeu *bataille des sexes*.

Pour ce type de jeu, il est montré que l'ensemble des équilibres de Nash existants est grand et que cette notion perd donc de son pouvoir prédictif. Plusieurs ***folk théorèmes***[7] viennent décrire les équilibres existants sur le long terme (Osborne et Rubinstein, 1994).

Dans le cas de la répétition du *dilemme du prisonnier*, malgré l'équilibre de Nash qui les pousse à *Dénoncer*, les joueurs ont intérêt à coopérer. S'ils le font, cette situation sera stable car chacun d'eux pense qu'en changeant de comportement, le gain immédiat sera annulé par la vengeance de l'adversaire qui va le *punir* (en le *dénonçant*) d'avoir mis fin de la coopération établie. Ce raisonnement n'est valable, que si l'adversaire *a le temps* de pouvoir se venger, donc quand le jeu n'a pas de limite en nombre de répétitions. On est dans une situation de **jeu répété à horizon infini**.

Cependant, dans les **jeux répétés à horizon fini**, la situation diffère. Même si un équilibre bénéfique est trouvé, un joueur peut, impunément, dévier lors de la dernière itération. Mais, en étant suffisamment loin de la fin, aucun joueur n'a d'intérêt à le faire (Osborne et Rubinstein, 1994).

7. dénomination donnée à ces théorèmes du fait qu'ils ont été manipulés par la communauté scientifique travaillant sur la théorie des jeux sans que la première personne à l'avoir fait ne puisse être désignée

1.5 Jeux évolutionnaires

La théorie des jeux évolutionnaires a été développée avec les biologistes dans le but de mieux comprendre le comportement animal et le phénomène de sélection naturelle.

Smith (1982), formalisant le concept central de **stratégie évolutionnairement stable** (ou *ESS*[8]), s'est rendu compte que la version évolutionnaire de la théorie des jeux est plus réaliste car elle n'exige pas que les joueurs agissent rationnellement mais uniquement qu'ils possèdent une stratégie bien définie. Les stratégies sont héritées génétiquement et peuvent potentiellement être modifiées par des mutations.

Comme pour l'équilibre de Nash, la propriété de stabilité évolutionnaire n'explique nullement comment la population arrive à obtenir la stratégie voulue. Au lieu de cela, la question est de savoir si, une fois entreprise, une stratégie est robuste face aux pressions évolutives (Weibull, 1997).

Afin d'évaluer sa capacité à faire survivre celui qui l'utilise, chaque stratégie est testée face à elle-même et face aux stratégies alternatives (en tenant compte de leur fréquence d'apparition dans la population globale).

Les processus évolutifs combinent deux mécanismes de base : la **mutation**, qui fournit la variété et la **sélection** qui favorise certaines variétés par rapport à d'autres en appliquant ce que l'on appelle la *dynamique du réplicateur* (expliquée en 1.5.2) (Weibull, 1997).

1.5.1 La stabilité évolutionnaire

La stabilité évolutionnaire étudie la réponse d'une population lors de l'introduction d'individus mutants. Concrètement, supposons que les individus d'une population donnée possèdent une stratégie s et qu'une proportion ϵ d'individus mutants (ayant comme stratégie s') y est introduite. Ces deux stratégies s'opposent dans un jeu symétrique (se caractérisant par le fait que $u_1(s,s') = u_2(s,s') = u(s,s')\ \forall s, s'$). C'est l'utilité moyenne obtenue par chaque groupe d'individus qui détermine la proportion des deux groupes dans la génération suivante. Cette utilité est donnée par les deux formules $U(s) = (1-\epsilon).u(s,s) + \epsilon.u(s,s')$ et $U(s') = (1-\epsilon).u(s',s) + \epsilon.u(s',s')$.

Pour que la stratégie s soit *évolutionnairement stable*, l'utilité des individus ayant la stratégie s doit être supérieure à l'utilité de ceux ayant la stratégie s'. Dans le cas contraire, les individus mutants seraient en mesure d'envahir la population. Ainsi, il faut que $U(s) > U(s'), \forall s'$. Selon Smith (1976), le nombre de mutants introduits étant petit (ϵ est pris comme étant négligeable), cette contrainte se traduit en :

$$\forall s',\ u(s,s) > u(s',s) \text{ ou } (u(s,s) = u(s',s) \text{ et } u(s,s') > u(s',s'))$$

8. *Evolutionary Stable Strategy*

Autrement dit, la stratégie s est un équilibre évolutionnaire si le profil de stratégies (s, s) est un *équilibre de Nash strict* $(\forall s', u(s, s) > u(s', s))$ ou s'il est un *équilibre de Nash faible* $(\forall s', u(s, s) \geq u(s', s))$ et que la stratégie s est plus efficace contre s' que s' l'est face à elle même $(u(s, s') > u(s', s'))$.

Soit l'exemple suivant −tiré de (Polak, 2007)− où une population d'individus interagissent en jouant au *dilemme du prisonnier*. Déterminons laquelle des deux stratégies pures possibles (*coopérer* (*i.e.* ne pas dénoncer) ou *dénoncer*−notées ici respectivement par c et d−) est évolutionnairement stable. D'abord, prenons une population de coopérateurs dans laquelle est introduite une proportion ϵ de mutants dénonciateurs. Les utilités moyennes des deux groupes sont :

$$U(c) = u(c, c).(1 - \epsilon) + u(c, d).\epsilon = -1.(1 - \epsilon) - 3.\epsilon = -1 - 2.\epsilon$$
$$U(d) = u(d, c).(1 - \epsilon) + u(d, d).\epsilon = 0.(1 - \epsilon) - 2.\epsilon = -2.\epsilon$$

Comme $U(d) > U(c)$, les individus mutants seront capables d'envahir la population. La stratégie c n'est donc pas évolutionnairement stable.

Prenons la situation inverse et voyons maintenant si les mutants coopérateurs vont être capables de s'introduire dans une population de dénonciateurs. On a :

$$U(d) = u(d, d).(1 - \epsilon) + u(d, c).\epsilon = -2.(1 - \epsilon) + 0.\epsilon = -2 + 2.\epsilon$$
$$U(c) = u(c, d).(1 - \epsilon) + u(c, c).\epsilon = -3.(1 - \epsilon) - 1.\epsilon = -3 + 2.\epsilon$$

De même, $U(d) > U(c)$. Les individus mutants coopérateurs ne seront pas capables d'envahir la population. Conclusion : la stratégie *dénoncer* est évolutionnairement stable. En fait, remarquer que cette stratégie est l'unique meilleure réponse des joueurs (l'équilibre de Nash est alors dit *strict*) nous aurait permis d'arriver directement à cette même conclusion.

Dans la nature, il arrive souvent que des animaux coopèrent entre eux, ce qui est contradictoire avec l'*ESS* prédit par l'analyse présentée ci-dessus. Comment la théorie des jeux explique-t-elle ce phénomène ? Une réponse possible consiste à ne pas se restreindre au modèle de reproduction asexuée (où les stratégies passent telles quelles d'une génération à l'autre). La prise en compte de la reproduction sexuée, peut certes, compliquer fortement le modèle, mais donne des résultats plus proches de ce que l'on observe dans la nature. Il est également possible de considérer un jeu plus élaboré où la coopération serait plus attrayante (comme un jeu répété) au lieu d'un simple jeu sous forme normale (Polak, 2007).

1.5.2 La dynamique du réplicateur

Notons la proportion d'individus de la population suivant la stratégie s (respectivement s') par p_s (resp. $p_{s'}$). Posons $\overline{U}$ comme étant l'utilité moyenne de la population globale, nous obtenons (Alexander, 2002) :

$$U(s) = p_s.u(s,s) + p_{s'}.u(s,s')$$
$$U(s') = p_s.u(s',s) + p_{s'}.u(s',s')$$
$$\overline{U} = p_s U(s) + p_{s'} U(s')$$

Supposons ensuite que la proportion d'individus suivant chaque stratégie dans la prochaine génération $(t+\delta t)$ est liée aux proportions d'individus de la génération actuelle (t) par les deux règles suivantes :

$$p_s(t+\delta t) = p_s(t)\frac{U(s)}{\overline{U}} \text{ et } p_{s'}(t+\delta t) = p_{s'}(t)\frac{U(s')}{\overline{U}}$$

Règles pouvant être réécrites ainsi :

$$p_s(t+\delta t) - p_s(t) = p_s(t)\frac{U(s)-\overline{U}}{\overline{U}}$$
$$p_{s'}(t+\delta t) - p_{s'}(t) = p_{s'}(t)\frac{U(s')-\overline{U}}{\overline{U}}$$

Si les proportions de chaque stratégie ne changent pas beaucoup d'une génération à l'autre, ces règles peuvent être approchées par les équations différentielles suivantes :

$$\frac{dp_s}{dt} = \frac{p_s(U(s)-\overline{U})}{\overline{U}}, \frac{dp_{s'}}{dt} = \frac{p_{s'}(U(s')-\overline{U})}{\overline{U}}$$

Ces équations différentielles, dites du **réplicateur** (*Replicator Equations*), montrent le taux de croissance de la proportion d'individus utilisant chacune des stratégies existantes. Le mot *réplicateur* fait référence aux stratégies qui se transmettent à la génération suivante en fonction de leur *fitness* (de leur utilité) dans la génération actuelle (Taylor et Jonker, 1978).

Poursuivons l'exemple de la section précédente et analysons la dynamique évolutive de la population ayant une proportion p_c de coopérateurs et p_d de dénonciateurs. On a donc $p_c > 0, p_d > 0, p_c + p_d = 1$ et :

$$U(c) = u(c,c).p_c + u(c,d).p_d = -1.p_c - 3.p_d$$
$$U(d) = u(d,c).p_c + u(d,d).p_d = 0.p_c - 2.p_d$$

On retrouve bien que $U(d) > U(c)$. Or, $\overline{U} = p_c U(c) + p_d U(d)$, donc $U(d) > \overline{U} > U(c)$. Ainsi, $\frac{U(c)-\overline{U}}{\overline{U}} < 0$ et $\frac{U(d)-\overline{U}}{\overline{U}} > 0$. D'où, $\frac{dp_c}{dt} = \frac{p_c(U(c)-\overline{U})}{\overline{U}} < 0$ et $\frac{dp_d}{dt} = \frac{p_d(U(d)-\overline{U})}{\overline{U}} > 0$.

La conclusion est que pour tout mélange de dénonciateurs et de coopérateurs, la *dynamique du réplicateur* induira une croissance des premiers et une décroissance des seconds jusqu'à atteindre l'état d'équilibre évolutionnaire ne possédant que de dénonciateurs.

1.6 Jeux coopératifs

Dans la théorie des jeux coopératifs, l'accent est mis sur ce que peuvent faire les groupes (ou *coalitions*) de joueurs une fois constitués. La formation de groupes suppose souvent que la coalition des joueurs engendre un gain supplémentaire par rapport à la somme

des gains individuels. Le gain du groupe doit pouvoir être partagé entre les membres de la coalition. C'est le cas lorsque l'utilité est *transférable* entre les joueurs (par exemple quand les gains sont des valeurs monétaires). Cela simplifie l'analyse des jeux car il suffit d'affecter une unique valeur d'utilité à chaque coalition (Leyton-Brown et Shoham, 2008).

Un jeu coopératif à utilité transférable est défini par le tuple (N, v) avec :

— N, l'ensemble de tous les joueurs.

— La fonction caractéristique $v : 2^N \to \mathbb{R}$, associant à chaque coalition $S \subseteq N$ une utilité qui vérifie $v(\emptyset) = 0$. Le 2^N représente l'ensemble de toutes les coalitions de joueurs possibles.

Selon Leyton-Brown et Shoham (2008), la théorie des jeux coopératifs cherche essentiellement à répondre aux questions suivantes :

— Quelles sont les coalitions qui vont se former ? Aura-t-on une *grande coalition*[9] ?

— Comment chaque coalition devra diviser son gain parmi ses membres ?

1.6.1 La valeur de Shapley

Shapley (1952) propose une distribution équitable de l'utilité de la grande coalition entre ses membres. L'idée est que les membres reçoivent une utilité proportionnelle à leur contribution marginale. Pour définir l'équité, Shapley propose les axiomes suivants :

— L'**axiome de symétrie** prévoit que les agents interchangeables devraient recevoir la même utilité $\psi : \psi_i(N, v) = \psi_j(N, v)$. On dit que les agents i et j sont **interchangeables** s'ils apportent la même valeur à toute coalition. $\forall S \subseteq N, i \notin S, j \notin S, v(S \cup \{i\}) = v(S \cup \{j\})$.

— Un joueur est dit **dummy**, si ce qu'il apporte à la coalition est égal à ce qu'il serait capable de réaliser seul ou $\forall S \subseteq N, i \in S, V(S \cup \{i\}) - V(S) = v(\{i\})$. **L'axiome du joueur dummy** dit que les joueurs *dummy* devraient recevoir une utilité égale aux gains qu'ils seraient capables d'obtenir en jouant seuls.

— Si on considère deux jeux coalitionnels différents, définis par deux fonctions caractéristiques différentes v_1 et v_2, impliquant le même ensemble d'agents, ***l'axiome d'additivité*** stipule que dans le cas où on modélise la situation comme un seul jeu dans lequel chaque coalition S réalise un gain de $v_1(S)+v_2(S)$, les utilités des agents de chaque coalition devraient être la somme des utilités qu'ils auraient atteint pour cette coalition dans les deux jeux distincts.

Il existe une unique distribution des gains satisfaisant les trois axiomes. Cette distribution a pour valeur ce qui est appelée la **valeur de Shapley** donnée par la formule :

9. Une grande coalition est un arrangement dans lequel tous les joueurs s'allient.

$$\psi_i(N,v) = \frac{1}{N!} \sum_{S \in N \setminus \{i\}} |S|!(|N| - |S| - 1)![v(S \cup \{i\}) - v(S)]$$

Cette expression capture la *contribution marginale moyenne* de l'agent i (notée ψ_i), en calculant la moyenne de toutes les différentes séquences selon lesquelles la grande coalition pourrait être construite à partir de la coalition vide. Dans une séquence d'ajouts, au moment où l'agent i est ajouté, sa contribution est $[v(S \cup \{i\}) - v(S)]$. Cette quantité est multipliée par les $|S|!$ différentes façons avec lesquelles l'ensemble S aurait pu être formé avant l'agent i et par les $(|N| - |S| - 1)!$ différentes manières avec lesquelles les agents restants pourront être ajoutés par la suite. La somme des contributions marginales dans tous les ensembles S possibles est divisée par $|N|!$ (le nombre d'ordres possibles de tous les agents) pour obtenir la contribution marginale moyenne (Leyton-Brown et Shoham, 2008).

1.6.2 Le *Cœur* (*The Core*)

Au vu de la division des utilités, les agents peuvent ne pas vouloir former la grande coalition. Il se peut que certains joueurs préfèrent former des coalitions plus petites, même si celles-ci conduisent à un gain global plus faible.

Une division motivera les agents à former une grande coalition si, et seulement si, elle appartient à l'ensemble ***Cœur***, définit ainsi (Leyton-Brown et Shoham, 2008) : Dans un jeu coopératif (N, v), une division donnant le vecteur de gains x est dans le *Cœur*, si et seulement si $\forall S \subseteq N, \sum_{i \in S} x_i \geq v(S)$.

Autrement dit, une division des utilités est dans le *Cœur*, si et seulement si aucun groupe n'a intérêt à se détacher de la grande coalition.

Dans le cas général, plusieurs divisions des utilités peuvent être dans le *Cœur* comme celui-ci peut être vide. De nombreux résultats permettent de prédire les propriétés du *Cœur*. Donnons d'abord les définitions suivantes :

— On dit qu'un jeu $G = (N, v)$ est un *jeu simple* si $\forall S \subseteq N, v(S) \in \{0, 1\}$.

— On dit qu'un joueur i est un joueur *véto* si $v(N \setminus \{i\}) = 0$.

— On dit qu'un jeu $G = (N, v)$ est *convexe* si

$$\forall S \subset N, \forall T \subset N, v(S \cup T) \geq v(S) + v(T) - v(S \cap T)$$

Parmi les résultats les plus intéressants concernant le *Cœur*, Leyton-Brown et Shoham (2008) citent :

— Dans un jeu simple, le *Cœur* est vide si, et seulement si, il n'existe aucun joueur véto. Dans le cas contraire, le *Cœur* est formé de toutes les fonctions de division qui n'affectent rien à tout joueur non *veto*.

— Tout jeu convexe possède un *Cœur* non vide.

— Dans un jeu convexe, la valeur de Shapley est dans le *Cœur*.

1.7 Applications de la théorie des jeux

Le grand degré d'abstraction des modèles de la théorie des jeux fait que de nombreuses applications pratiques l'utilisent : Les études sur le *Cœur* permettent de comprendre la stabilité des prix sur les marchés (selon l'offre, la demande, l'arrivée de nouveaux venus ou la formation de coalitions) (Osborne et Rubinstein, 1994). La théorie des jeux est utilisée pour étudier et concevoir différents types de ventes aux enchères. Nous citons par exemple le *GSP auction* [10] (Edelman et al., 2005) qui est utilisé, entre autres, par *Google* et *Yahoo* dans leur système publicitaire de paiement par clic.

Dans le contexte des réseaux informatiques, la théorie des jeux aide à concevoir des mécanismes qui amènent les participants (ici les nœuds du réseau) à se comporter comme souhaité par le concepteur . Les réseaux *pair-à-pair* sont souvent modélisés en utilisant la théorie des jeux (Srikant, 2015) étant donné que les ressources du réseau sont partagées et que chacun souhaite les exploiter au maximum. D'autres problématiques, telles que le routage ou la congestion constituent aussi des domaines d'application (Roughgarden, 2002; Hoang et al., 2015). Dans les *WMN*, des méthodes d'affectation de fréquences sont basées sur des modélisations issues de la théorie des jeux (Duarte et al., 2012).

Dans les réseaux ad hoc, il n'existe pas d'infrastructure de routage. Au lieu de cela, chaque nœud achemine les paquets de ses voisins. Ces nœuds peuvent être détenus par des entités différentes potentiellement en concurrence. La théorie des jeux a été utilisée dans le cadre de la conception (1) de protocoles d'accès au canal réduisant les interférences (Lai et al., 2012) et (2) de mécanismes poussant les nœuds égoïstes à acheminer le trafic (autrement ils ne participeraient pas au routage afin de conserver leur énergie) (Charilas et Panagopoulos, 2010; Hoang et al., 2015).

Dans le domaine des radios cognitives, les utilisateurs secondaires tentent d'exploiter les bandes de fréquences non utilisées par leurs propriétaires. Cette exploitation peut être réalisée après détection de l'utilisation du spectre. Pour plus de fiabilité, il est dans l'intérêt des utilisateurs secondaires de partager entre-eux les informations concernant le spectre utilisable. Cependant, ces utilisateurs étant en concurrence, ils peuvent décider de diffuser des fausses informations ou de ne rien diffuser. La théorie des jeux est utilisée pour étudier la formation de coalitions de partage d'information (Hossain et Han, 2009). Elle est également utilisée pour concevoir des mécanismes qui motivent les joueurs à partager des informations correctes (Hoang et al., 2015).

Les mécanismes de la théorie des jeux sont aussi utilisés dans les réseaux à infrastructure comme les réseaux sans fil locaux (*WLAN* [11]) et les réseaux cellulaires pour aider à créer des mécanismes d'accès au canal qui assurent la coopération des appareils mobiles et conduisent à une bonne qualité de service (Hoang et al., 2015). D'autres utilisations

10. *Generalized Second-Price auction*
11. *Wireless Local Area Networks*

de la théorie des jeux dans les réseaux cellulaires sont la minimisation des interférences entre les stations de bases et les *Femtocells*[12] (Wang et al., 2015) et l'étude des avantages du partage des bandes de fréquences entre différents opérateurs (Kamal et al., 2009).

Conclusion

La théorie des jeux permet la modélisation et la résolution de situations où les individus (plus ou moins) rationnels interagissent pour satisfaire, au mieux, leurs propres intérêts. Les différents types de jeux permettent de représenter des situations réelles et offrent une base théorique utilisable que ce soit du point de vue du joueur ou de celui du concepteur du jeu. La recherche, en théorie des jeux, vise, de plus, l'amélioration des méthodes de calcul des différents concepts de solution qui dans certains cas sont difficiles à déterminer.

La relative abstraction des notions manipulées par la théorie des jeux fait que de nombreux domaines d'application, autres que l'économie, s'approprient de plus en plus ses concepts et ses idées.

L'optimisation combinatoire fait partie des domaines qui cherchent à intégrer la théorie des jeux dans la variété de méthodes déjà à sa disposition. Aussi, le chapitre suivant aborde justement les approches proposées par la théorie des jeux pour l'optimisation multi-objectif.

12. Les Femtocells sont des petites stations de base domestiques qui se connectent au réseau du fournisseur de service (opérateur téléphonique) via une connexion filaire et assurent la couverture réseau dans les zones où le signal de l'antenne-relais est faible ou inexistant

Chapitre 2

La théorie des jeux pour l'optimisation multi-objectif

Introduction

La théorie des jeux et l'optimisation multi-objectif sont, de par leur nature même, étroitement liées. Elles partagent, en effet, bon nombre de notions mathématiques. Il arrive que la première ait besoin de la seconde, dans le cas de recherche des solutions de concept, à titre d'exemple. Nous nous intéressons, néanmoins, ici, à l'inverse : c'est à dire aux apports et aux contributions de la théorie des jeux pour la résolution de problèmes d'optimisation multi-objectif, et ce, en vue d'une application au problème d'affectation de fréquences.

Par souci de clarté, nous décrivons, d'abord, certaines notions essentielles concernant l'optimisation multi-objectif. Le lecteur notera, à juste titre, la grande ressemblance avec la sous-section 1.1.4. Nous citerons ensuite, les métriques utilisées pour la comparaison des algorithmes multi-objectifs (dont nous aurons besoin au chapitre 6). Nous donnons ensuite deux exemples d'algorithmes multi-objectifs. Ceux-ci étant basés sur les algorithmes génétiques nous décriront le principe de fonctionnement de ces derniers. Enfin, nous énumérerons, sans être exhaustifs, différents travaux que nous avons distingués en approches basées sur l'équilibre de Nash, sur des hybridations Nash-Pareto en plus des approches coopératives et co-évolutionnaires.

2.1 Notions d'optimisation multi-objectif

L'optimisation multi-objectif (ou multicritère, appelée aussi optimisation vectorielle) se démarque de l'optimisation mono-objectif de par le nombre de critères qui sont poursuivis en même temps.

Un problème d'optimisation multi-objectif est défini ainsi :

$$\begin{cases} min\ F(x) = (f_1(x), f_2(x), ..., f_n(x)) \\ avec\ x = (x_1, x_2, ..., x_k) \in C \end{cases}$$

où $n \geq 2$ désigne le nombre d'objectifs, x le vecteur ayant comme composantes les k variables de décision, $F(x)$ le vecteur à optimiser et C l'ensemble des solutions réalisables résultat de différent types de contraintes (d'égalité ou d'inégalité) (Meunier, 2002; Periaux et al., 2015). La *minimisation* peut être remplacée, d'une manière équivalente par une *maximisation*.

Il est souvent possible de se placer dans l'ensemble $Y = F(C)$ qui représente l'ensemble des points objectifs réalisables. Cet ensemble est muni d'une relation d'ordre partielle appelée *relation de dominance* (Meunier, 2002). Du fait qu'il est rare de pouvoir trouver un vecteur arrivant à optimiser, simultanément, tous les objectifs (certains peuvent être contradictoires et des compromis doivent être trouvés), une autre approche est alors utilisée pour considérer qu'une solution est optimale. Il s'agit de la Pareto-optimalité.

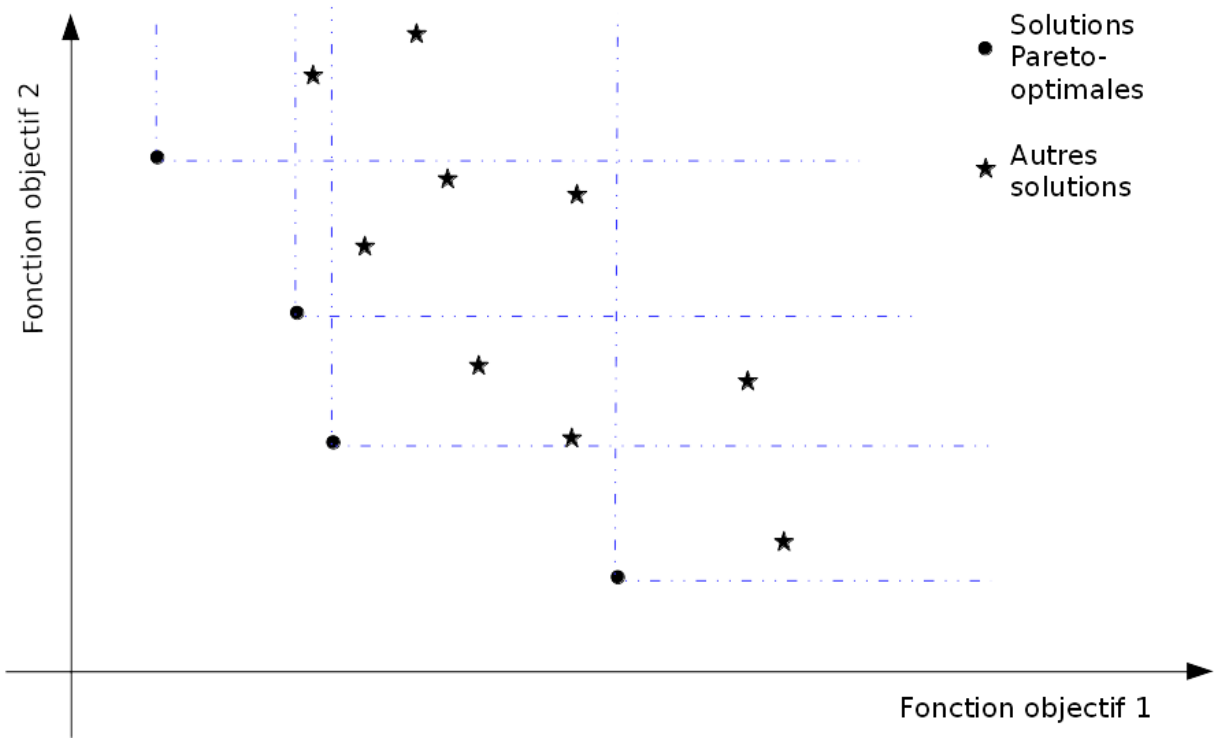
Une solution $Y = (y_1, y_2, ..., y_n)$ **domine**[1] une solution $Y' = (y'_1, y'_2, ..., y'_n)$, si et seulement si $\forall i \in [1...n]\ y_i \leq y'_i$ et $\exists i \in [1...n]$ tel que $y_i < y'_i$. Une solution $x^* \in C$ est dite **Pareto-optimale** si et seulement si, il n'existe pas de solution $x \in C$ telle que $F(x)$ domine $F(x^*)$ (Meunier, 2002). Dans la figure 2.1, les deux axes représentent les fonctions objectifs à minimiser. Les points ronds sont de solutions Pareto-optimales ; elles dominent les points en étoile. Le but d'un algorithme de résolution sera donc de trouver, faute d'optimum global, une (ou plusieurs) solution(s) Pareto-optimale(s).

Une approche classique consiste à transformer le problème afin de devenir mono-objectif (avec une combinaison linéaire des différents objectifs par exemple). L'inconvénient est la perte d'information (due au rassemblement des contraintes) et la perte de représentativité des objectifs. Plusieurs autres méthodes de transformation existent et le lecteur intéressé peut se référer à (Meng et al., 2010) pour plus d'approfondissement.

Nous nous intéresserons, ici, surtout aux méthodes qui ne cherchent pas forcément à réduire le nombre de fonctions objectifs. Dans ce contexte, nous retrouvons, à travers la littérature, une utilisation fréquente des algorithmes génétiques ou d'autres méthodes évolutionnaires manipulant une population de solutions. Ces méthodes permettent, en effet, de trouver, d'un coup, plusieurs solutions Pareto-optimales au lieu d'effectuer une série d'exécutions. De plus, ces méthodes sont moins sensibles à la répartition géométrique des solutions, appelée *frontière de Pareto*, alors que pour les méthodes mathématiques classiques la présence de discontinuités, par exemple, dans cette frontière, est problématique (Coello, 2001).

1. Rappelons que nous sommes dans le cadre d'une minimisation

FIGURE 2.1 – Graphe illustrant le concept de la Pareto optimalité dans le cas de deux fonctions objectifs à minimiser.



2.2 Comparaison des algorithmes d'optimisation multi-objectif

Le fait que la relation de dominance au sens de Pareto ne soit pas une relation d'ordre totale fait qu'il n'est pas trivial de comparer différents algorithmes multi-objectifs traitant du même problème, ni même différentes exécutions du même algorithme avec des paramètres variées. De plus, ses algorithmes fournissent un front contenant plus qu'une seule solution. Comparer deux algorithmes revient donc à comparer deux fronts de Pareto ou à leur affecter une certaine métrique de qualité. Différentes métriques ont été proposées. Elles différent selon qu'elles soient destinées à évaluer la qualité (on parle de métriques **absolues**) ou à comparer les algorithmes (on parle de métriques relatives) (Collette et Siarry, 2002).

Zitzler et al. (2000) présentent une métrique relative C permettant de comparer deux fronts F' et F'' et retournant un réel dans $[0, 1]$.

$$C(F', F'') = \frac{card(\{x'' \in F''; \exists x' \in F' |\ x' \leq x''\})}{card(F'')}$$

Le symbole $\leq$ spécifie la relation de couverture : $x' \leq x''$ (Lu : le vecteur x' couvert x'') signifie que soit $x' < x''$ (x' domine x'') ou bien que $x' = x''$. Plus $C(F', F'')$ est proche de 1, plus les solutions de F'' sont couvertes par F'. Néanmoins, il est toujours nécessaire

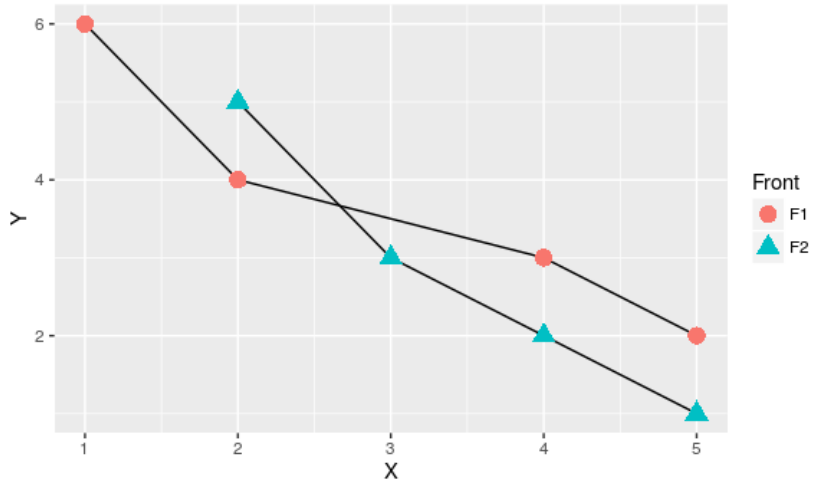


FIGURE 2.2 – Illustration de deux fronts de solutions

C(ligne, colonne)	F1	F2
F1	1	1/4
F2	1/2	1

TABLEAU 2.1 – Comparaison entre les deux fronts de solutions de la figure 2.2

de calculer également $C(F'', F')$ car la somme des deux valeurs n'est pas toujours égale à 1.

Dans la figure 2.2, deux fronts de solutions sont présentés. Aucun ne domine entièrement l'autre mais $F2$ domine plus souvent $F1$ que le contraire. C'est ce qu'indique la métrique C dans le tableau 2.1. La lecture se fait colonne par colonne. Le front ayant les valeur les plus petites dans sa colonne (seules les valeurs hors diagonale sont prises en considération) est considéré comme étant le meilleur.

Les auteurs présentent également des métriques absolues d'évaluation de la qualité des fronts trouvés. Il s'agit des fonctions M_2 et M_3 qui doivent être maximisées : La fonction M_2 mesure le nombre de $\sigma-$niches (c.à.d : Le nombre de solutions qui n'ont pas de voisins situés à une distance inférieure à σ). Elle donne un nombre réel dans l'intervalle $[0, card(F)]$.

$$M_2(F) = \frac{1}{1 - card(F)} \times \sum_{x \in F} card(\{y \in F, ||x - y|| > \sigma\})$$

La fonction M_3 mesure l'étendue de la surface du front en prenant la somme des plus grandes distances sur chaque composantes des solutions.

$$M_3(F) = \sqrt{\sum_{i=1}^{n} \max(||x_i - y_i||; x, y \in F)}$$

Elle ne donne pas d'indication sur la qualité intrinsèque des solutions générées mais permet d'en évaluer la variété. D'autres métriques existent mais elles nécessitent la connaissance de la vrai frontière de Pareto (les solutions optimales du problème) ce qui n'est pas possible dans le cas de problèmes pratiques.

2.3 Les algorithmes génétiques multi-objectifs

La présence de multiples objectifs dans un problème donne lieu à un ensemble de solutions optimales (connues sous le nom de solutions Pareto-optimales). En absence d'informations supplémentaires, aucune de ces solutions ne peut être dite meilleure que les autres. Cela exige de trouver autant de solutions Pareto-optimales que possible. Quelques auteurs ont suggéré la conversion du problème en un problème mono-objectif en mettant l'accent sur un objectif particulier à un moment donné. Une telle méthode doit être appliquée à plusieurs reprises en espérant de trouver une solution différente à chaque exécution. D'autres auteurs ont suggéré d'utiliser des algorithmes génétiques multi-objectif, la principale raison étant la capacité de ces algorithmes à trouver de multiples solutions Pareto-optimales en une seule exécution. Par ailleurs, le mécanisme de sélection peut être conçu de sorte à maintenir un ensemble diversifié de solutions avec une pression évolutionnaire qui dirige la recherche vers la région Pareto-optimale (Deb et al., 2002).

Les algorithmes génétiques *NSGA-II* et *SPEA2* sont parmi les meilleurs algorithmes d'optimisation multi-objectif. Avant de les décrire, commençons par une description des mécanismes régissant le fonctionnement des algorithmes génétiques dans le cas général.

2.3.1 Les algorithmes génétiques

Dans son livre "*Adaptation in natural selection and artificial systems*", Holland (1975) a formalisé un plan (ou mécanisme) d'adaptation inspiré de la biologie, et capable de conduire à des structures de qualité en un temps raisonnable. Ce mécanisme est connu sous le nom *d'algorithmes génétiques* ou *évolutionnaires*. Les algorithmes génétiques ont été utilisé avec un grand succès pour trouver de bonnes solutions aux problèmes d'optimisation combinatoire.

L'hypothèse de base de la sélection naturelle est que les êtres vivant les plus adaptés à leur environnement ont plus de chance de voir leurs propriétés génétiques se perpétuer dans les générations suivantes. Ces propriétés (ou traits) sont encodées par les gènes contenus dans les cellules de l'individu sous forme de chromosomes. Par ailleurs, il est remarquable que la plupart des organismes les plus complexes se reproduisent de sorte à ce que les descendants reçoivent un mélange de matière génétique des deux parents. Il en résulte que les combinaisons des différents traits des parents rentrent en compétition.

Les algorithmes génétiques s'inspirent de ces phénomènes naturels : Les solutions des

problèmes de recherche combinatoire sont modélisées comme des individus d'une population en compétition et sont représentées par des chaînes de bits appelées chromosomes. Pour quantifier la qualité des individus, on définit une fonction d'évaluation (ou de fitness) qui associe à chaque individu (chromosome) une valeur numérique proportionnelle à son niveau d'adaptation.

Les algorithmes génétiques simulent la sélection naturelle : Un processus itératif est exécuté où les nouveaux individus sont construits par combinaison de ceux de la génération précédente avec un biais en faveur des individus de bonne qualité. Ce processus se répète jusqu'à la satisfaction d'un critère d'arrêt (Comme par exemple : Après un certain nombre de générations). La structure d'un algorithme génétique est donné par l'algorithme 1.

Algorithme 1 Structure de base d'un algorithme génétique (Srinivas et Patnaik, 1994)

```
1: procédure ALGORITHME GÉNÉTIQUE ()
2:     Générer une population initiale ;
3:     Évaluer la population ;
4:     tant que La condition d'arrêt n'est pas satisfaite faire
5:         Sélectionner des individus qui vont participer à la reproduction ;
6:         Appliquer les opérateurs de croisement et de mutation pour générer la progéniture ;
7:         Évaluer la population ;
8:     fin tant que
9: fin procédure
```

Pour pouvoir appliquer l'algorithme génétique à un problème concret, il faut, au préalable, définir :

- **Le codage génétique d'une solution potentielle** ;
- **La fonction d'évaluation (ou de fitness)** ;
- **La méthode de génération de la population initiale** : qui peut être soit aléatoire soit basée sur des heuristiques ;
- **Les opérateurs génétiques (croisement et mutation)**,
- **Le mécanisme de sélection**,
- et d'autres considérations comme le choix de la **taille de population**, de la **condition d'arrêt** et de la **politique du remplacement**.

A Opérateurs de croisement

Le rôle de l'opérateur de croisement (ou de ***crossover***) est de combiner l'information génétique de deux chromosomes parents pour produire des chromosomes fils. Les croisements les plus connus dans la littérature sont (Bessedik et al., 2014) :

— **Le croisement à un point** : Un unique point de croisement sur les deux chromosomes parents est choisi au hasard. Tous les gènes au-delà de ce point sont échangés entre les deux parents. Le résultat étant les deux chromosomes enfants.

— **Le croisement à deux points** : Deux points sont aléatoirement sélectionnés sur les chromosomes parents. Tous les gènes entre les deux points sont échangées entre les parents, donnant les deux chromosomes enfants.

— **Le croisement à N points** : C'est une généralisation des deux premiers. N points sont choisis, au hasard, sur les parents, résultant en $N+1$ segments. Les $(2k)^{ème}$ segments sont échangés et les $(2k+1)^{ème}$ ne le sont pas.

— **Le croisement uniforme** : Tout gène dans les 2 chromosomes parents est séparément considéré pour être échangé avec une probabilité donnée par le concepteur.

Dans certaines représentations génétiques, les opérateurs de croisement présentés ci-dessus peuvent générer des chromosomes invalides. Par exemple, si le chromosome contient une permutation de nombres, l'application du croisement à un point sur deux permutations valides peut générer des permutations invalides :
Parent 1 : 4 6 7 2 | 3 1 8 5
Parent 2 : 7 1 4 8 | 5 2 3 6
Fils 1 : 4 6 7 2 | **5 2 3 6** (Répétition de 2 et 6 et absence de 1 et 8)
Fils 2 : 7 1 4 8 | **3 1 8 5**

Pour remédier à ce problème, des opérateurs de croisement spécialisés on été développés. Nous citons en guise d'exemple (Valenzuela, 2001) :

— **Partially matched crossover (PMX)** : Les deux parents sont alignés et 2 positions de coupe sont choisies uniformément. Ces deux points définissent une section d'échange où pour chaque parent, chaque gène est échangé par celui qui lui correspond dans l'autre parent. En appliquant cet opérateur sur les deux chromosomes de l'exemple précédent, les nombre 7, 4 et 8 dans les deux chromosomes fils sont échangés par 4, 8 et 5 respectivement :
Parent 1 : 4 6 | 7 2 3 | 1 8 5
Parent 2 : 7 1 | 4 8 5 | 2 3 6
Fils 1 : **7** 6 | **4 8 5** | 1 **2 3**
Fils 2 : **4** 1 | **7 2 3** | **8 5** 6

— **Cycle crossover (CX)** : On identifie des cycles entre les valeurs des deux parents. Ensuite, on construit la progéniture de sorte à ce que le fils 1 récupère les gènes des cycles de rangs pairs à partir du parent 2 et les gène des cycles impaires à partir du parent 1. Le fils 2 est le complémentaire du fils 1. Pour l'exemple précédent, le premier cycle est $4 \to 7 \to 4$ et le deuxième est $6 \to 1 \to 2 \to 8 \to 3 \to 5 \to 6$. Les fils générés sont donc :

Fils 1 : 4 **1 7 8 5 2 3 6**
Fils 2 : 7 **6** 4 **2 3 1 8 5**

Dans l'algorithme génétique de Holland, l'opérateur de croisement n'est pas appliqué à tous les parents. Après le choix des deux chromosomes parents, l'algorithme génère un nombre aléatoire entre 0 et 1. L'opérateur de croisement est invoqué quand ce nombre est inférieur à un seuil spécifié par l'utilisateur appelé **probabilité de croisement** noté p_c.

B Opérateurs de mutation

Après le croisement, les chromosomes fils sont soumis à une mutation. La forme la plus simple de mutation est de changer la valeur d'un bit de 0 à 1 ou vice versa. Tout comme p_c contrôle la probabilité d'un croisement, un autre paramètre, le taux de mutation p_m, désigne la probabilité de basculement d'un bit. Cette probabilité doit être faible, sinon, la recherche va se transformer en une recherche aléatoire. Les gènes d'un chromosome sont mutés de façon indépendante. Autrement dit, la mutation d'un gène ne modifie pas la probabilité de mutation des autres.

Le rôle de la mutation est la restauration du matériel génétique perdu (Srinivas et Patnaik, 1994). Par exemple : Supposons que toutes les chaînes dans une population ont convergé à un 0 à une position donnée et que la solution optimale possède un 1 à cette position. Le croisement ne peut pas régénérer de 1 à cette position tandis qu'une mutation pourrait le faire.

Comme pour le croisement, certains codages chromosomiques nécessitent des opérateurs de mutations spécialisés. Parmi ceux développés pour les chromosomes contenant des permutations, on cite (Valenzuela, 2001) :

— **Position based mutation** : Cette mutation consiste à sélectionner deux nombres au hasard et à placer le deuxième immédiatement avant le premier dans la liste.

— **Order based mutation** : Deux nombres sont sélectionnés au hasard, et leur position sont interverties.

C Mécanismes de sélection

Au début de chaque génération, l'algorithme génétique doit sélectionner des individus de la population (potentiellement avec répétition) pour qu'ils soient soumis aux croisements et aux mutations. Cette sélection doit être biaisée en faveur des individus de bonne qualité.

Parmi les mécanismes de sélection les plus connus, citons (Srinivas et Patnaik, 1994) :

— **Fitness proportionate selection** (Sélection proportionnelle à la fitness) : La probabilité qu'un individu soit choisi est égale au rapport de sa fitness (valeur à maximiser) sur la somme des fitness des individus de la population.

— **Rank based selection** (Sélection proportionnelle au rang) : La probabilité qu'un individu soit choisi est proportionnelle à son classement (ou rang) dans la population.

— **Tournament selection** (Sélection par tournoi) : k chromosomes sont choisis au hasard. Le meilleur parmi eux est sélectionné.

Les deux derniers mécanismes présentent l'avantage de remédier aux deux principaux problèmes de la sélection proportionnelle à la fitness (Srinivas et Patnaik, 1994) : 1) Dans les premières générations de l'algorithme génétique, la population a généralement une valeur de fitness moyenne faible. La présence de quelques gènes de fitness relativement élevées (appelés *superstrings*) font que le mécanisme de sélection tend à allouer un grand nombre de descendants à ces *superstrings*, ce qui conduit à une convergence prématurée. 2) Dans les étapes ultérieures de recherche, lorsque la population a convergé et que la variance des valeurs de fitness est devenue petite, le mécanisme de sélection proportionnelle alloue un nombre approximativement égal de descendants à tous les chromosomes, réduisant ainsi le favoritisme envers les meilleurs individus.

2.3.2 Exemples d'algorithmes génétiques pour l'optimisation multi-objectif

Nous exposons ici les algorithmes *NSGA-II* et *SPEA2* qui figurent parmi les méthodes d'optimisations multi-objectif évolutionnaires les plus performantes :

A Non-dominated Sorting Genetic Algorithm 2 (*NSGA-II*)

NSGA-II (Deb et al., 2002) est un algorithme génétique multi-objectif élitiste[2]. Il classe la population en différents niveaux de non-domination. Les individus de chaque niveau sont indifférents entre eux (c.à.d qu'aucun individu ne Pareto-domine un autre) mais les individus d'un niveau dominent tous les individus des niveaux supérieurs. La figure 2.3 illustre cela dans le cas de deux fonctions objectifs.

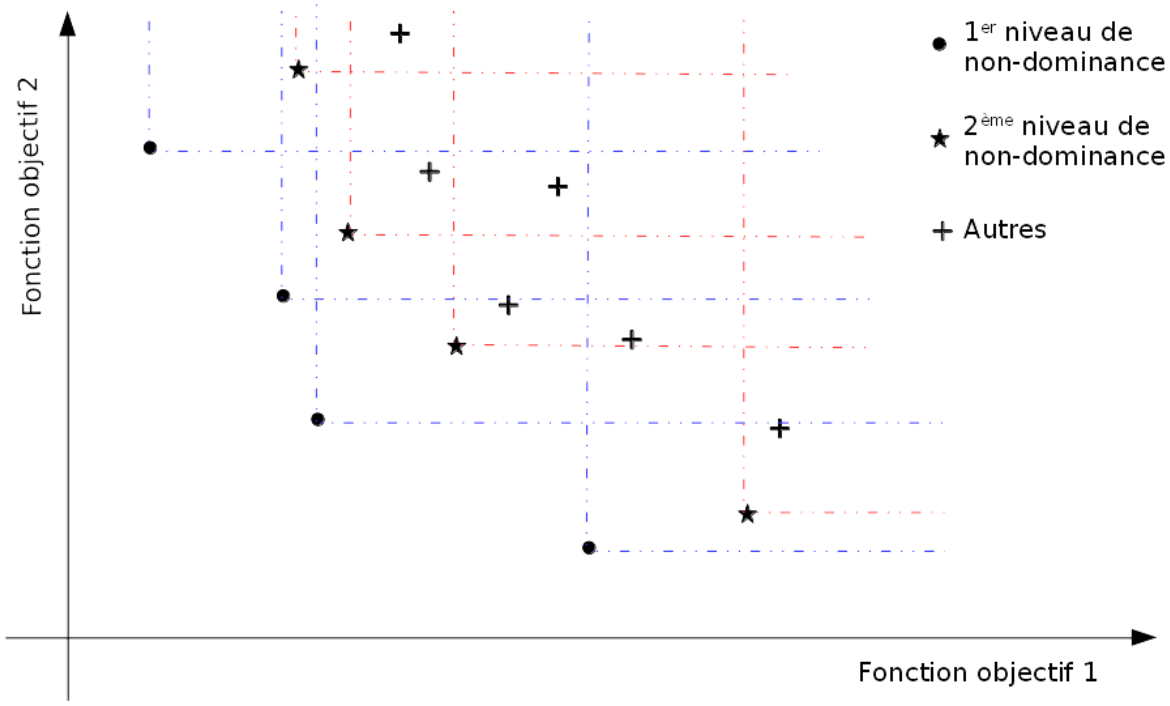
Pour maintenir un bon niveau de diversification, *NSGA-II* trie les solutions appartenant à chaque niveau dans l'ordre croissant de la densité de leur entourage. La densité des solutions entourant une solution particulière dans un niveau est estimée par l'inverse de la distance moyenne des deux solutions les plus proches dans chacun des objectifs. L'algorithme 2 détaille ce calcul.

Étant donné le rang de chaque solution (i_{rank}) (le numéro du niveau de non-domination la contenant) et la distance moyenne de ses solutions voisines ($i_{distance}$), l'opérateur de comparaison entre deux solutions i et j est défini comme suit : $i < j$ (i est meilleur que j) si $i_{rank} < j_{rank}$ ou ($i_{rank} = j_{rank}$ et $i_{distance} > j_{distance}$).

La boucle principale de *NSGA-II* est illustré par l'algorithme 3.

2. Par élitiste, on veut dire que l'algorithme conserve les solutions de bonne qualité

FIGURE 2.3 – Graphe visualisant les niveaux de non-domination.



Algorithme 3 L'algorithme *NSGA-II*

1: **procédure** L'ALGORITHME *NSGA-II* ()
2: Générer une population initiale P de taille N ;
3: Trier la population par dominance au sens de Pareto ;
4: **tant que** La condition d'arrêt n'est pas satisfaite **faire**
5: Effectuer une sélection par tournoi binaire et appliquer les opérateurs de croisement et de mutation pour générer une population Q ;
6: Trier $P \cup Q$ (en utilisant l'opérateur de comparaison défini) ;
7: $P \leftarrow$ les premières N solutions de $P \cup Q$;
8: **fin tant que**
9: **fin procédure**

B Strength Pareto Evolutionary Algorithm 2 (*SPEA2*)

SPEA2 (Zitzler et al., 2001) est un algorithme génétique multi-objectif élitiste. Il maintient une archive de solutions non dominées. La valeur de fitness est donnée sur la base de la dominance Pareto. Afin de diversifier la recherche, les solutions qui sont indifférentes, vis à vis de la relation de dominance, sont différenciées en attribuant des valeurs de fitness supérieures aux solutions les plus éloignées de ses voisins (dans l'espace des fonctions objectifs).

La fitness des solutions est calculée par l'algorithme 4 où N et la taille de la population

Algorithme 2 L'algorithme qui calcule l'estimation de la distance moyenne des solutions autour d'une solution donnée

procédure ESTIMATION DES DISTANCES MOYENNES (I UN NIVEAU DE NON-DOMINATION)
    $l \leftarrow |I|$;
    pour $i \in [1, l]$ **faire**
        $I[i]_{distance} \leftarrow 0$;
    fin pour ;
    pour chaque objectif m **faire**
        Trier I selon les valeurs d'évaluation données par l'objectif m ;
        $I[1] \leftarrow I[l] \leftarrow \infty$;
        $f_m^{min} \leftarrow I[1]$; (La valeur minimale de la fonction objectif m)
        $f_m^{max} \leftarrow I[l]$; (La valeur maximale de la fonction objectif m)
        pour $i \in [2, l-1]$ **faire**
            $I[i]_{distance} \leftarrow I[i]_{distance} + (I[i+1]_m - I[i-1]_m)/(f_m^{max} - f_m^{min})$;
        fin pour
    fin pour
fin procédure

est $\bar{N}$ celle de l'archive.

Algorithme 4 Calcule des valeurs de fitness dans *SPEA2*

procédure ÉVALUER (POPULATION)
    $K \leftarrow \sqrt{N + \bar{N}}$
    Établir les relations de dominance entre toutes les solutions ;
    Construire la matrice des distances entre toutes les solutions ;
    pour chaque solution i **faire**
        $S(i) \leftarrow$ le nombre de solution que i domine ;
        $j \leftarrow$ le $K^{ème}$ voisin le plus proche dans l'espace des objectifs ;
        $D(i) \leftarrow 1/(2 + distance(i, j))$;
    fin pour
    pour chaque solution i **faire**
        $R(i) \leftarrow$ la somme des $S(j)$ des individus j dominant i ;
        $Fitness(i) \leftarrow R(i) + D(i)$
    fin pour
fin procédure

L'archive est peuplée par les solutions non Pareto-dominées. Si la taille de l'archive est inférieure à $\bar{N}$, on y ajoute les solutions dominées de bonnes valeurs de fitness. Si la taille de l'archive est supérieure à $\bar{N}$, une procédure de troncature est exécutée : La solution présentant la distance minimale vis à vis d'une autre est choisie à chaque étape. S'il y a plusieurs solutions avec une distance minimum, la deuxième plus petite distance est considérée et ainsi de suite. Cette procédure est exécutée jusqu'à ce que la taille de l'archive soit égale à $\bar{N}$. L'algorithme *SPEA2* est donné par l'algorithme 5.

Algorithme 5 L'algorithme *SPEA2*

procédure L'ALGORITHME *SPEA2* ()
  $P \leftarrow$ une population initiale de taille N ;
  $A \leftarrow \emptyset$;
  tant que La condition d'arrêt n'est pas satisfaite **faire**
    Évaluer $(A \cup P)$;
    $A \leftarrow$ les solutions non dominées de $A \cup P$;
    si $|A| < \bar{N}$ **alors**
      Ajouter des solutions dominées à A ;
    fin si
    si $|A| > \bar{N}$ **alors**
      Appliquer l'opérateur de troncature à A ;
    fin si
    Effectuer une sélection par tournoi binaire et appliquer les opérateurs de croisement et de mutation sur les éléments de A pour générer une nouvelle population P ;
  fin tant que
fin procédure

2.4 Travaux d'optimisation multi-objectif basés sur la théorie des jeux

2.4.1 Travaux basés sur l'équilibre de Nash

Sefrioui et Perlaux (2000) ont créé le concept d'algorithme génétique de Nash (ou *Nash Genetic algorithm*). Cette approche est une modification de l'algorithme génétique dont le but est de trouver un équilibre de Nash dans un cadre multi-objectif. Le processus d'optimisation est modélisé par un jeu : pour N objectifs, le chromosome (une représentation d'une solution du problème −comme décrit au point 2.3) est segmenté en N parties avec la création de N **joueurs**. Chaque joueur i possède une population et optimise la $i^{ème}$ fonction objectif (f_i) en manipulant le $i^{ème}$ segment du chromosome. L'espace des **stratégies** d'un joueur est l'ensemble de valeurs possibles sur son segment du chromosome. Un chromosome complet représente **une issue** possible du jeu où chaque joueur a choisi une stratégie. **Le profit** capturé par chaque joueur est calculé en évaluant le chromosome complet par sa propre fonction objectif. Ce processus est décrit par l'algorithme 6 dans lequel, $Population_i$ représente l'ensemble de stratégies du joueur i et $Meilleur_i$ représente sa meilleure stratégie.

Cette approche ressemble à l'approche dite *par îlots* des algorithmes génétiques à la différence qu'il y a ici plusieurs objectifs.

Clarich et al. (2003) proposent un algorithme multi-objectif basé sur la théorie des jeux, et en particulier sur les jeux non-coopératifs et la *méthode de Nelder-Mead*. Cette dernière optimise un seul objectif en utilisant, pour n variables, $n + 1$ solutions en un nombre m d'itérations. À chaque étape, le point le plus mauvais (selon l'évaluation de

Algorithme 6 L'algorithme *Nash GA*

procédure L'ALGORITHME *Nash GA*()
    Le chromosome est segmenté en N partie $x_1, ..., x_N$.
    pour i allant de 1 à N **faire**
        $Meilleure_i \leftarrow$ une stratégie aléatoire ;
    fin pour
    pour i allant de 1 à N **faire**
        $Population_i \leftarrow$ un ensemble de stratégies aléatoires ;
        Évaluer $Population(i)$;
    fin pour
    pour i allant de 1 à N **faire**
        $Meilleure_i \leftarrow \underset{S \in Population_i}{argmin} \; Fitness(S)$;
    fin pour
    tant que La condition d'arrêt n'est pas satisfaite **faire**
        pour i allant de 1 à N **faire**
            $Sélection(Population_i)$;
            $Croisement(Population_i)$;
            $Mutation(Population_i)$;
            Évaluer $Population(i)$;
            $Tronquer(Population_i, taille\, population)$;
        fin pour
        pour i allant de 1 à N **faire**
            $Meilleure_i \leftarrow \underset{S \in Population_i}{argmin} \; Fitness(S)$;
        fin pour
        pour i allant de 1 à N **faire**
            Évaluer $Population(i)$;
        fin pour
    fin tant que
    retourner $concaténation(Meilleure_i, i \in [1, N])$
fin procédure
procédure ÉVALUER POPULATION(i)
    pour chaque stratégie $S \in Population_i$ **faire**
        $Stratégies\, des\, autres \leftarrow concaténation(\{Meilleur_j, j \neq i\})$;
        $issue \leftarrow concaténation(S, Stratégies\, des\, autres)$;
        $Fitness(S) \leftarrow f_i(issue)$;
    fin pour
fin procédure
procédure TRONQUER ($Population$, N)
    Trier $Population$ selon les valeurs de $Fitness$
    Supprimer les stratégies au delà des N premières
fin procédure

l'objectif) est remplacé par son symétrique par rapport à l'hyperplan formé par les autres points. Cette méthode s'apprête plus aux méthodes d'optimisation à variables réelles.

Xiao et al. (2015) ont analysé la similarité entre le domaine de la théorie des jeux et celui de l'optimisation multidisciplinaire multi-objectif et ont énuméré des modélisations reliant les deux domaines, à savoir, l'approche Pareto (ou coopérative), celle de Nash (ou non-coopérative) et l'approche de Stackelberg (ou *leader/follower*). Ils ont proposé une nouvelle approche pour les problèmes d'optimisation multi-objectif dans des environnements non-coopératifs basés sur la programmation de l'expression des gènes (*GEP* [3]) et l'équilibre de Nash. La méthode de *GEP* est utilisée afin de simuler les actions rationnelles.

2.4.2 Travaux basés sur une hybridation Nash-Pareto

Lee et al. (2010) ont proposé une hybridation de l'algorithme génétique multi-objectif *NSGA-II*. L'idée est d'avoir un joueur par objectif pour la recherche de l'équilibre de Nash en coopération avec un joueur dit *Pareto* (qui, lui, va exécuter *NSGA-II*), comme décrit dans l'algorithme 7.

3. *Gene Expression Programming* : algorithme évolutionnaire (génome / phénome) où les individus se présentent sous deux formes : des chaînes linéaires de longueur fixe (gènes ou chromosomes) au cours de la reproduction et des entités non linéaires de différentes formes et tailles (phénotype) pendant l'évaluation de la qualité de la solution

Algorithme 7 L'algorithme de Lee et al. (2010)

Entrées: Soient x et y les deux variables du problème avec les deux fonctions objectifs f_1 et f_2. Le joueur 1 (J1) prend la variable x et la fonction f_1. Le joueur 2 (J2) prend la variable y et la fonction f_2, tandis que le joueur Pareto (JP) peut manipuler les deux variables.

1: Initialisation des 3 populations (une par joueur) ;
   - JP initialise un population aléatoirement
   - JP transmet la meilleure valeur de y (y_{elite}) à J1.
   - J1 initialise une population aléatoire (avec la partie y fixée à y_{elite}).
   - J1 transmet la meilleure valeur de x (x_{elite}) à J2.
   - J2 initialise une population aléatoire (avec la partie x fixée à x_{elite}).
   - JP trie sa population en utilisant l'algorithme *NSGA-II*.

2: **tant que** condition d'arrêt non satisfaite **faire**

3: Effectuer une sélection par tournoi dans les trois populations :
   - Pour tous les individus de la population de J1, avec $f_1(concaténation(x, y_{elite}))$.
   - Pour tous les individus de la population de J2, avec $f_2(concaténation(x_{elite}, y))$.
   - Pour JP, sélection *NSGA-II* (selon les classements et la distance inter-individus)

4: Croisement et mutation sur les individus sélectionnés.

5: Échange des meilleures solutions :
   - J1 transmet l'individu $concatenation(x_{elite}, y_{elite})$ à JP. x_{elite} étant la meilleure valeur de x trouvée jusqu'à présent et y_{elite} la dernière valeur reçue de y.
   - J2 transmet l'individu $concatenation(x_{elite}, y_{elite})$ au joueur Pareto. y_{elite} étant la meilleure valeur de y trouvée jusqu'à présent et x_{elite} la dernière valeur reçue de x.
   - J1 transmet la valeur x_{elite} trouvée au Le J2.
   - J2 transmet la valeur y_{elite} trouvée au Le J1.

6: Tri et troncature de la population de JP (avec l'algorithme de *NSGA-II*).

7: **fin tant que**

2.4.3 Travaux basés sur les jeux coopératifs

Li et al. (2009) ont proposé une approche basée sur les essaims particulaires. Il s'agit d'un essaim de particules (une par solution) positionnées aléatoirement au départ et qui sont déplacées suivant une certaine vélocité. Contrairement aux travaux précédents, où la population est divisée avec un objectif pour chaque sous-population, ici les auteurs considèrent des agent intelligents externes à la population capables de la guider. Au début chacun d'eux tire de son coté mais, à l'approche des solutions recherchées, il y a coopération afin d'éviter les blocages et d'obtenir des solutions de meilleure qualité.

2.4.4 Travaux basés sur les jeux (co-)évolutionnaires

Les jeux co-évolutionnaires sont des extensions des algorithmes évolutionnaires classiques qui se distinguent du fait que la *fitness*[4] des individus (provenant de plusieurs populations) est calculée sur la base de leurs interactions avec les individus des autres populations.

Sim et al. (2004) ont développé un algorithme co-évolutionnaire non-coopératif pour l'optimisation bi-objectif. Leur algorithme maintient deux populations. Afin d'évaluer les individus, chaque individu x est confronté à un individu x' de l'autre population. La *fitness* est calculée comme suit[5] : $Fitness(x) = (f_1(x) - f_2(x'))$ et $Fitness(x') = (f_2(x') - f_1(x))$, f_1 et f_2 étant les fonctions objectifs à minimiser. Cette approche a démontré de très bons résultats comparables à ceux trouvés par les premières versions de *SPEA* et *NSGA* (décrits en 2.3.2).

Antonio et Coello (2015) ont utilisé une approche de type équilibre de Nash pour éviter la convergence prématurée des algorithmes co-évolutionnaires. Plus spécifiquement, il s'agit dans leur travail de perfectionner l'algorithme génétique co-évolutionnaire coopératif (*CCGA*[6]) proposé par Potter et Jong (1994) qui améliore l'algorithme génétique standard (pour des problèmes de décision avec 30 variables de décision par exemple), mais souffre de convergence prématurée. L'idée du *CCGA* est de diviser le problème en sous composantes et de faire évoluer celles-ci d'une manière coopérative. L'amélioration de Antonio et Coello vient du fait que toutes les combinaisons entre individus (à l'intérieur et entre les populations) sont évaluées et les meilleures sont trouvées grâce à l'équilibre de Nash. Un joueur est, ici, un groupement d'individus dont l'intérêt est porté sur une fonction objectif bien défini.

2.4.5 Discussion

Les différentes approches citées ont des points communs : presque toutes se basent sur les opérateurs utilisés dans les algorithmes génétiques et presque toutes emploient un joueur par fonction objectif. Néanmoins elles se basent sur des concepts différents de la théorie des jeux ou tentent de les trouver de manières différentes (pour rappel, l'équilibre de Nash est équivalent à l'équilibre évolutionnaire). L'algorithme 7 reprend presque tout les principes de l'algorithme 6 en rajoutant des améliorations.

Il est notable que les approches de (Sefrioui et Perlaux, 2000; Clarich et al., 2003; Sim et al., 2004; Antonio et Coello, 2015) cherchent l'équilibre de Nash ou l'équilibre co-évolutionnaire alors que ceux-ci n'offrent aucune garantie vis à vis de la qualité des solutions générées. Un autre point faible de ces approches est qu'elles n'emploient aucun

4. Terme désignant la qualité des solutions, surtout dans le contexte évolutionnaire.
5. Si on supprime le facteur de normalisation
6. *Cooperative Coevolutionary Genetic Algorithm*

mécanisme pour la conservation des solutions de bonne qualité contrairement à l'approche de Lee et al. (2010) où une population exécute l'algorithme élitiste *NSGA-II*.

Conclusion

La théorie des jeux a été, de multiples façons, employée dans la perspective d'amélioration des méthodes d'optimisation multi-objectif. Diverses approches, basées sur des jeux, coopératifs et non-coopératifs ont été expérimentées. La plupart des méthodes retournent une population de solutions dites *Pareto-optimales*. De ce fait, l'utilisation de méthodes basées sur les algorithmes génétiques est naturel. En effet, leurs caractéristiques font que de nombreux travaux basés sur la théorie des jeux les emploient. Nous avons choisi de faire de même pour la résolution du problème d'affectation de fréquences, qui, lui, est l'objet du chapitre suivant.

Chapitre 3

Le problème d'affectation de fréquences

Introduction

Les réseaux de téléphonie mobiles dits GSM [1] sont des réseaux de télécommunication de deuxième génération, succédant ainsi aux précédents systèmes analogiques. De nouvelles normes (dont la norme UMTS [2] dite de troisième génération) sont apparues par la suite afin de répondre aux besoins grandissants en débit, notamment pour l'accès Internet.

Ces technologies ont connu, et continuent à connaître, un grand essor avec des taux de pénétration très élevés. Ainsi, l'industrie de la téléphonie mobile affichait plus de 3,6 milliards d'utilisateurs uniques fin 2014 (soit environ la moitié de la population mondiale) et un milliard d'utilisateurs supplémentaires sont prévus d'ici 2020. L'écosystème mobile s'impose comme un acteur majeur de l'économie mondiale. Rien qu'en 2014, il a généré 3,8% du *Produit Mondial Brut (PMB)* avec une contribution de 3000 milliards de dollars répartie sur 236 pays (GSM Association, 2015).

Afin de satisfaire leurs clients tout en minimisant leurs coûts, les opérateurs (les compagnies de téléphonie) se doivent de porter une attention particulière à la conception et à la configuration de leurs infrastructures. De plus, la demande croissante pose continuellement des défis d'ingénierie dont la bonne résolution conditionne le respect des engagements techniques et commerciaux des opérateurs, notamment en terme de *qualité du service* offerte au client (ce qui comprend aussi bien la disponibilité du réseau que la non interruption des communications établies).

L'un des défis centraux lancés par les réseaux mobiles est le problème d'affectation de fréquences (ou PAF). Aussi, nous nous concentrerons sur les réseaux de téléphonie mobiles tout en mentionnant d'autres situations où la problématique existe.

Dans ce chapitre, nous allons décrire le contexte de ce problème, qui s'applique pour le GSM et l'UMTS, donner sa définition formelle ainsi que les modélisations, les variantes

1. *Global System for Mobile Communications*
2. *Universal Mobile Telecommunication System*

et les méthodes de résolution existantes tout en mettant l'accent sur sa difficulté de résolution. Nous citerons aussi, avec plus de détails, différentes approches de résolution basées sur les algorithmes génétiques.

3.1 Contexte

Le réseau mobile GSM a pour objectif de permettre aux appareils mobiles d'accéder au réseau téléphonique, via une liaison électromagnétique. Il était conçus, à l'origine, pour le transport de la voix, tout comme le réseau téléphonique filaire. D'autres services sont venus, ensuite, se greffer sur son infrastructure, à l'image du SMS [3] pour la messagerie ou du WAP [4] pour l'accès Internet. La croissance de la demande, notamment induite par l'augmentation du trafic de données multimédia, a conduit à la création de nouvelles normes de téléphonie cellulaires telle l'UMTS.

C'est l'organisme appelé *3GPP* (*3rd Generation Partnership Project*), regroupant les principaux organismes de standardisation mondiaux de télécommunication, qui est chargé d'établir les recommandations de la troisième et de la quatrième génération. Ses travaux ont, notamment, donné naissance aux protocoles *HSDPA* et *HSUPA* [5] qui définissent ce que l'on appelle la 3G+ avec comme évolution la définition des systèmes de quatrième génération dits *LTE-A* [6] (Coupechoux et Martins, 2013). Toutes ces avancées technologiques restent, néanmoins, basées sur la communication par ondes électromagnétiques (ou radio communication)

La caractéristique principale de toute onde est sa fréquence. Les limitations de la physique et de la technologie font que la gamme de spectre utilisable va de 3 KHz à 300 GHz environ. La disponibilité du spectre de fréquences au niveau mondial est régulée par l'ITU [7]. Les gouvernements se chargent, ensuite, de contrôler l'exploitation du spectre par les différents opérateurs auxquels des licences d'utilisation d'une ou de plusieurs bandes de fréquences sont accordées. Ces bandes peuvent être, ou ne pas être, homogènes sur l'ensemble de l'étendue géographique couverte. Une bande du spectre électromagnétique disponible $[f_{min}, f_{max}]$ est généralement partitionnée en un ensemble de canaux de même largeur (donc de même bande passante) Δ. Ces canaux fréquentiels sont notés $F = \{1, ..., N\}$ avec $N = (f_{max} - f_{min})/\Delta$. Nous verrons plus loin que, toutes ces fréquences peuvent ne pas être disponibles partout (Eisenblätter et al., 2002; Aardal et al., 2007).

Les déploiements les plus communs des réseaux sans fils sont : Les réseaux basés sur des infrastructures (comme les réseaux cellulaires précités), les réseaux ad hoc (trouvés

3. *Short Message Service*
4. *Wireless Application Protocol*
5. *High Speed Uplink/Downlink Packet Access*
6. *Long Term Evolution Advanced*
7. *International Telecommunication Union*

généralement dans les réseaux de capteurs) et les réseaux cognitifs. Audhya et al. (2011) citent de nouveaux modèles de réseaux sans fil ayant émergé récemment et pour lesquels d'autres types de problèmes d'affectation de fréquences se posent. Nous les reprenons à titre indicatif :

A Les réseaux Mesh

Les réseaux *Mesh* ou *WMN* (*Wireless Mesh Networks*) constituent un développement alternatif récent aux réseaux cellulaires et ad hoc. Ils sont constitués de deux types de nœuds : *passerelles* et *non passerelles*. Les *WMN* routent les paquets des nœuds *non passerelles* via les nœuds *passerelles*. L'intérêt de ce type de réseau est de permettre une extension significative de la zone de couverture. Les *WMN* sont aussi plus sûrs, vu l'existence de plusieurs chemins possibles vers les nœuds *passerelles*. Cette sûreté est un aspect critique lors de catastrophes naturelles par exemple. Cependant, l'affectation de fréquences devient encore plus délicate avec la décentralisation du réseau et l'augmentation du nombre de liens et par conséquent du taux d'interférence.

B Les réseaux de radios cognitives

Face à l'explosion du nombre d'appareils sans fils opérant dans la bande de fréquence libre (non allouée), les bandes louées sont, elles, sous-utilisées. La commission américaine des communications (FCC[8]) a préconisé alors le développement d'une nouvelle génération de radios qui fonctionneraient comme des utilisateurs secondaires dans les bandes de fréquences louées sans interférer avec les utilisateurs primaires. Ces radios, dites *cognitives*, sont capables de mesurer l'utilisation du spectre afin de l'utiliser plus efficacement. Dans ce contexte, l'affectation de fréquences doit tenir compte de l'égoïsme potentiel des équipements. Les mécanismes de la théorie des jeux sont souvent employés pour assurer une équité entre les utilisateurs secondaires.

3.2 Les réseaux de téléphonie mobile cellulaires

Nous décrivons ici la structure d'un réseau cellulaire de type GSM ou UMTS (qui sont grossièrement les mêmes pour la partie découpage du réseau qui nous intéresse ici) mais en utilisant la terminologie propre au premier.

L'étendue géographique à couvrir est décomposée en secteurs appelés **cellules**. Une cellule est l'aire géographique couverte par une station de base et où le signal de la base est le plus fort.

Une **station de base** (ou ***BTS*** pour *Base Transeiver Station*) est associée à chaque cellule. La station abrite différentes antennes (entre une et quatre) ainsi que du matériel de

8. *Federal Communications Commission*

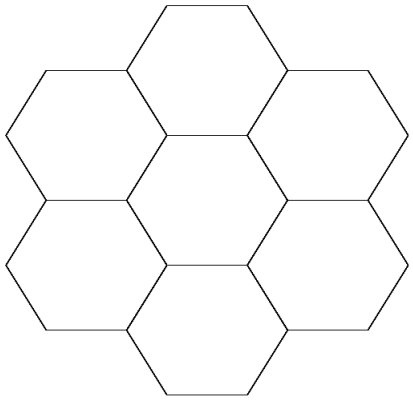


FIGURE 3.1 – Exemple de modélisation hexagonale d'un réseau cellulaire

traitement du signal. C'est via ces antennes que les communications sont établies avec les terminaux mobiles (les téléphones). Les différentes stations de base sont interconnectées (en général par liaison filaire) à un **centre de commutation** (ou ***MSC*** pour *Mobile Switching Centers*) qui les relie au réseau téléphonique fixe et à Internet (Hemazro et al., 2008). Le transport correct de la voix (ou d'une autre donnée) nécessite que le signal transmis soit de qualité et de puissance suffisantes. Cette puissance se dissipant avec la distance parcourue par l'onde, la bonne répartition des *BTS* est primordiale (Mahmoudi, 2011). A titre indicatif, dans la terminologie UMTS, on parle de **Node-B** au lieu de *BTS* (Coupechoux et Martins, 2013).

La subdivision du réseau en cellules permet la réutilisation de ressources spectrales (c'est à dire les fréquences d'émission) à différents emplacements du réseau. Néanmoins, il est nécessaire de séparer par une distance suffisante deux utilisations de la même fréquence ou de fréquences voisines. Grâce à la réutilisation des fréquences, un opérateur peut supporter un nombre de communications simultanées qui est largement supérieur au nombre de fréquences à sa disposition (Mahmoudi, 2011).

Les réseaux cellulaires sont souvent modélisés par une grille de cellules hexagonales (Aardal et al., 2007) ressemblant aux alvéoles d'une ruche d'abeilles, comme le montre la figure 3.1. Cette subdivision du réseau n'est, cependant, pas fidèle à la réalité. En effet, les opérateurs font en sorte que les cellules puissent se chevaucher pour ne pas interrompre les communications des utilisateurs se déplaçant de cellule en cellule. Le changement de cellule, appelé *hand-over* (ou *handoff*), est totalement invisible pour l'utilisateur. Le réseau est responsable de la gestion des éventuels déplacements des terminaux mobiles dans et entre les cellules (Mahmoudi, 2011).

3.2.1 Les interférences

Les signaux transmis doivent être clairs et compréhensibles à leur arrivée, afin d'être correctement restitués. Les bruits, provenant d'autres signaux émis à des fréquences interférentes, peuvent altérer leur bonne réception (Aardal et al., 2007). D'où la nécessité d'éviter au maximum les interférences.

La quantification des interférences se fait le plus souvent en mesurant le rapport entre le signal utile et le signal interférant au niveau du récepteur. On parle de *SIR* (pour *Signal to Interference Ratio*) (Kaci et Ait El Hara, 2011).

En général, le niveau d'interférence décroît rapidement avec la distance (en termes de nombre de canaux) séparant les fréquences concernées. Cependant, la puissance des signaux, leur fréquence même, leur direction ou encore le relief et les conditions météorologiques sont également des facteurs impactant le niveau d'interférence. Devant la difficulté de prédiction de ces facteurs et de l'impossibilité d'influer sur certains d'entre eux, une simplification largement adoptée consiste à ignorer les conditions environnementales, de supposer des antennes omnidirectionnelles (diffusant dans toutes les directions) et de ne se baser que sur la fréquence et la distance parcourue par l'onde pour le calcul de l'interférence (Aardal et al., 2007; Audhya et al., 2011). Les expérimentations ont montré que le niveau d'interférence reste faible, si les contraintes exposées ci-après sont respectées (Kumar Singh et al., 2012).

3.2.2 Contraintes imposées

En se basant sur le point précédent, on peut dégager trois types d'interférences qui formeront un ensemble de **contraintes de compatibilité électromagnétique** à satisfaire (Aardal et al., 2007; Audhya et al., 2011) :

1. **Contrainte co-cellule** (ou *co-cell separation constraint*) : des canaux affectés à une même station v doivent différer d'au moins une distance minimale (calculée en nombre de canaux) ;
2. **Contrainte co-site** (ou *co-site constraint CSC*) : deux canaux utilisés au sein d'une même cellule (par des antennes différentes u et v) doivent être séparés par une différence spectrale minimale ;
3. **Contrainte de canal adjacent** (ou *adjacent channel constraint ACC*) : des canaux voisins ne peuvent être affectés à certaines paires de cellules simultanément à moins que celles-ci ne soient géographiquement assez distantes ;
4. **Contrainte co-canal** (ou *co-channel constraint CCC*) : le même canal ne peut être réutilisé sur des cellules géographiquement proches.

Ces contraintes forment, avec la contrainte de **réponse à la demande** (stipulant que toute requête doit recevoir réponse), des **contraintes dures** qui doivent être satisfaites

(Kumar Singh et al., 2012).

À côté des contraintes dures, d'autres contraintes appelées **souples** (aussi appelées **duplex** (Bessedik, 2011)) sont prises en compte (Kumar Singh et al., 2012) :

— La contrainte de *packing* : stipulant qu'un canal utilisé soit réemployé le plus prés possible de l'emplacement de sa première utilisation tout en respectant les contraintes dures (ce qui minimise le nombre de canaux utilisés et par conséquent la probabilité de bloquer de futurs appels, faute de canal disponible) ;

— Un nombre minimum de changement doit être effectué en cas de réaffectation de canaux.

Un opérateur ne peut pas employer certaines fréquences au niveau des cellules se trouvant à la frontière du pays où il se trouve. Une entente avec les homologues avoisinants est nécessaire. On note souvent par $F(v)$ le sous-ensemble disponible de F pour une station de base v (Aardal et al., 2007). Les canaux non disponibles sont dits **bloqués** (Eisenblätter et al., 2002).

Résoudre le PAF revient donc à utiliser la bande de fréquence disponible pour répondre aux requêtes, en satisfaisant absolument les contraintes dures, tandis que les contraintes souples peuvent être violées malgré qu'elles permettent d'optimiser les ressources utilisées et d'améliorer la qualité de service offerte (Kumar Singh et al., 2012).

3.3 Définition du problème d'affectation de fréquences

Le problème d'affectation de fréquences est apparu dans les années 1960 avec l'avènement des premiers services de téléphonie sans fil. Hale (1980) précise qu'on parle d'*affectation de canaux* quand les fréquences allouées sont séparées par une différence fixe. Le terme *allocation* est parfois employé au lieu d'*affectation* ainsi que l'est le terme *canal* au lieu de *fréquence*. La littérature aborde, ainsi, souvent le problème *d'affectation de canaux* en l'appelant *affectation de fréquences* sachant qu'il est plus simple de faire abstraction des fréquences elles-mêmes (en ne considérant qu'un ensemble d'entiers représentant les fréquences) peu importe si elles sont dispersées uniformément ou non (Eisenblätter et al., 2002).

Comme décrit par Cotta et Troya (2001), la définition du PAF, comme pour tous les problèmes d'optimisation combinatoire, nécessite de donner une caractérisation d'une instance du problème, une caractérisation d'une solution et une mesure de qualité des solutions obtenues (c'est à dire une fonction objectif). Ainsi, une **instance** du PAF est un tuple $PAF(E, F, D, R, I)$ où :

— $E = \{e_1, ..., e_n\}$ est l'ensemble des émetteurs ;

— $F = \{f_1, ..., f_m\}$ est l'ensemble des fréquences disponibles ;

— f telle que $D(e, e')$ est la distance entre les émetteurs e et e' ;

— $R : E \to \mathbb{N}$ telle que $R(e)$ est le nombre de fréquences demandés par l'émetteur e ;

— $I : \mathbb{R} \to \mathbb{N}$ est une fonction telle que $I(d)$ est le nombre de fréquences de séparation nécessaires pour éviter une interférence entre deux émetteurs séparés d'une distance d.

Avec ces notations, une solution est un vecteur $\alpha = (\alpha_1, ..., \alpha_n) \in [2^F]^n$ (où chaque α_i est l'ensemble de fréquences affectée à e_i) vérifiant que :

— Les demandes sont totalement satisfaites donc $\forall e \in E, |\alpha_e| = R(e)$;

— Aucune paire d'émetteurs n'a de fréquences qui puissent interférer, ou

$$\forall e, e' \in E \not\exists f, f' \in F : f \in \alpha_e, f' \in \alpha_{e'}, |f - f'| < I(D(e, e'))$$

"Deux cas de figure se présentent pour la satisfaction des contraintes du PAF : un premier cas, où la satisfaction des contraintes d'interférences est relativement simple, il s'agit de chercher, parmi les solutions réalisables, celle qui convient le mieux ou qui coûte le moins (par exemple celle utilisant le moins de fréquences ou occupant un plus petit spectre). Le deuxième cas se présente quand la satisfaction des contraintes d'interférences est impossible."(Ghrissi, 2012) Les deux cas de figure sont gérés par différentes variantes du PAF présentés en section 3.5.

Selon Józefowicz et al. (2014), le PAF est applicable pour les réseaux GSM et UMTS. Ce n'est pas le cas avec les réseaux de type LTE-A qui permettent, en utilisant une méthode d'accès au canal avancée, l'utilisation de toutes les fréquences dans une même cellule (Elsner et al., 2015). Le problème est le même pour les réseaux de diffusion radio et de télévision. La seule véritable différence provient de la bande de fréquence utilisée. Celle-ci peut, néanmoins, ajouter des contraintes supplémentaires. Par exemple, dans la TV-UHF [9], le fait que la bande de fréquence inclut des harmoniques (des multiples) de fréquences de la même bande interdit l'affectation de fréquences dont la séparation est de 1, 2, 5, ou 14 (Aardal et al., 2007).

Il est à noter que le trafic de téléphonie mobile est bidirectionnel ; aussi deux fréquences doivent être choisies (une pour chaque sens de la communication). Celles-ci doivent être, bien entendu, éloignées pour éviter les interférences. Néanmoins le choix d'une seule fréquence est pris en compte dans la littérature. En effet, dans la pratique, on dispose de deux spectres de fréquences disjoints. Ce mécanisme simple permet de réduire la difficulté du choix des fréquences en employant la même répartition pour les deux bandes (Kaci et Ait El Hara, 2011).

9. Télévision Ultra Haute Fréquence

3.4 Modélisation par la coloration de graphe

C'est à Metzger (1970) (tel que cité par Hale (1980)) qu'est attribuée l'idée d'utiliser les méthodes mathématiques d'optimisation, et notamment la coloration de graphe, pour la résolution du PAF. Jusqu'aux années 1980, la plupart des approches étaient basées sur des heuristiques liées au *problème de coloration de graphe* comme l'indique Hale (1980).

Les premières modélisation du problème d'affectation de fréquences utilisaient la coloration de graphe, des variantes telle la **T-coloration** (initiée par Hale (1980) et permettant de prendre en considération les interférences du canal adjacent, alors que jusque-là on ne considérait que les contraintes co-canal) ont permis d'améliorer sensiblement la qualité des solutions (Aardal et al., 2007). Dans la T-coloration, les couleurs sont représentées par des entiers. La différence entre deux entiers de sommets partageant une arête ne doit pas appartenir à un ensemble T.

De multiples modèles existent pour le PAF ; leurs différences dépendent essentiellement des caractéristiques du réseau, de la modélisation des différents types d'interférences et de l'objectif fixé (Dupont et al., 2008).

On représente habituellement le PAF par un graphe $G = (V, E)$ appelé *graphe d'interférence* ou *graphe de contrainte*. Chaque station de base correspond à un sommet $v \in V$. Si deux sommets v et w peuvent interférer, alors une arête est créée ($vw \in E$). Le fait qu'une seule antenne requiert rarement une unique fréquence peut être modélisé en créant autant de copies que nécessaire du sommet concerné. Cette façon de faire, causant une explosion de la taille du graphe, est le plus souvent délaissée au profit d'une représentation plus compacte travaillant avec la multiplicité des demandes directement au niveau des sommets.

Afin de gérer la non disponibilité de toutes les fréquences, la *coloration par liste* à été introduite. Ce qui, couplée à la *T-coloration*, a donné la *T-coloration par liste*. Alors que, jusque là, une seule couleur était affectée par nœud, on a traité la multiplicité des demandes d'une station au niveau d'un seul sommet réduisant la taille totale du graphe. Plusieurs couleurs sont assignées à un même nœud (tout en respectant la contrainte co-site) dans ce qui est appelé la *T-coloration par ensembles* (Eisenblätter et al., 2002).

Il est possible, pour chaque paire de fréquences allouées $f \in F(v)$ et $g \in F(w)$, de quantifier le niveau d'interférence par une pénalité notée $p_{vw}(f, g)$. Comme expliqué précédemment, la pénalité ne dépendant que de l'emplacement des sommets et de la différence spectrale entre les fréquences, elle est, donc, une fonction de v, w et $|f - g|$ uniquement. À partir de cette représentation, deux alternatives existent dans la littérature. Soit on introduit une distance minimale d_{vw} telle que $p_{vw}(f, g)$ n'existe que si $|f-g| < d_{vw}$; soit on ne modélise que les contraintes co-canal (donc quand $|f - g| = 0$) et de canal adjacent (donc quand $|f - g| = 1$) (Aardal et al., 2007).

Le plus souvent ces pénalités, représentées dans une matrice dite *de pénalité*, ne sont

prises en considération que si elles dépassent un certain seuil p_{max} représentant le *SIR* maximum admis. L'intérêt est de réduire la taille du problème et de le formuler comme un simple *problème de satisfaction de contrainte* (CSP [10]). Dans le cas où $p_{vw}(f, g)$ ne dépend que de la distance $|f - g|$, on obtient, en utilisant le seuil p_{max}, un ensemble de distances interdites : ce qui est équivalent au problème de *T-coloration* où les fréquences deviennent alors des couleurs. Le problème consistant à savoir si les contraintes peuvent être satisfaites est appelé **PAF de faisabilité** ou *F-FAP* [11] (Aardal et al., 2007).

Pour guider le choix des solutions plusieurs métriques ont été proposées. Les différentes métriques utilisées font la diversité des variantes du PAF.

3.5 Variantes

Pour la résolution du PAF, il est possible de se restreindre à une optimisation mono-objectif (en ne considérant qu'une seule des variantes présentées ici) ou alors utiliser une optimisation multi-objectif en combinant deux ou plusieurs variantes.

3.5.1 Affectation de spectre minimum *MS-FAP* (*Minimum Span FAP*)

Dans le cas de la variante *MS-FAP*, l'opérateur est censé payer pour l'ensemble des fréquences comprises entre la plus petite et la plus grande fréquence utilisées (qu'elles soient toutes employées ou pas). Le spectre utilisé doit donc être minimisé. L'avantage de cette formulation est que, pour une demande donnée, il est possible de satisfaire l'ensemble des contraintes d'interférences en utilisant peu de fréquences. Néanmoins, même avec des méthodes d'optimisation puissantes, il reste difficile de trouver une affectation optimale (Audhya et al., 2011).

3.5.2 Affectation d'ordre minimum *MO-FAP* (*Minimum Order FAP*)

Dans son article, Hale (1980) montre, sur différents exemples simples, que les solutions obtenues en minimisant le nombre de fréquences utilisées sont meilleures ou aussi bonnes que celles obtenues en minimisant le spectre. Donc, du fait que les instances réelles sont beaucoup plus complexes que les petits exemples illustratifs qu'il donne, Hale affirme qu'il est judicieux de minimiser le nombre de fréquences utilisées. Ceci a donné naissance à la variante dite d'ordre minimum.

10. *Constraint Satisfaction Problem*
11. *Feasible-FAP*

3.5.3 Affectation d'interférence minimum *MI-FAP* (*Minimum interference FAP*)

Lorsque le spectre alloué est fixe, il est plus intéressant pour l'opérateur de minimiser le niveau d'interférences engendré par l'affectation (Hemazro et al., 2008). De plus, les deux modèles précédents utilisent la forme simplifiée (par seuillage) de la matrice de pénalité. Une manière d'utiliser complètement les informations de cette matrice est de considérer la somme des pénalités ou la pénalité maximale introduite par une affectation (le choix revenant à l'opérateur) (Eisenblätter et al., 2002). La variante *MI-FAP* présente une plus grande difficulté de résolution que les deux premières alternatives (Aardal et al., 2007).

3.5.4 Affectation à spectre fixe (*Fixed Spectrum FAP*)

La formulation *à spectre fixe* considère un spectre maximal autorisé B (qui peut être plus petit que celui retourné par le *MS-FAP*). Une fonction de coût, dépendant de B, est minimisée (elle peut représenter le nombre de contraintes violées ou le nombre d'appels bloqués par exemple) . Cependant, dans le cas d'instances difficiles, il est pratiquement impossible de réduire à zéro cette fonction de coût avec B fréquences (Audhya et al., 2011) ce qui réduit le champ d'application de cette variante.

3.5.5 Affectation de perturbation minimale *PM-FAP* (*Perturbation-Minimizing FAP*)

Cette variante se base sur l'aspect dynamique (décrit en 3.7) qui caractérise les problèmes réels. On se donne une affectation initiale des fréquences pour l'ensemble du réseau puis on reçoit une nouvelle requête. Celle-ci peut être due, soit, à un nouvel appel, à un *hand-over* ou à une fin d'appel. L'objectif est d'adapter la situation du réseau à ces petits changements tout en limitant le nombre de modifications de l'affectation établie et en garantissant la qualité des communications (Audhya et al., 2011).

3.5.6 Affectation de service maximum (*Max-FAP*) et Affectation de blocage minimum

Dans le cas où le *F-FA*P ne peut être résolu, on peut accepter une solution partielle en répondant à un maximum de demandes possibles. On tente d'assurer un service maximum. Pour ce faire, la contrainte dure de réponse à toutes les requêtes est levée. Jaumard et al. (2002) ont observé que cette variante a tendance à n'allouer que peu de fréquences à certains des sommets *difficiles* tandis que d'autres sommets sont complètement satisfaits. Pour assurer une qualité de service plus uniforme, Jaumard et al. ont introduit une borne inférieure $l(v)$ sur le nombre de fréquences à allouer par sommet.

Une autre solution consiste à minimiser la probabilité de blocage au niveau des sommets en fonction du nombre de fréquences déjà allouées. Ce que l'on appelle l'affectation de blocage minimum (Aardal et al., 2007).

3.6 Difficulté du problème

La résolution du problème affectation de fréquence se réduit à celle du problème de coloration de graphe classée comme étant $\mathcal{NP}-$difficile (Hale, 1980). Ce qui enlève tout espoir de trouver une solution optimale en temps polynomial (en terme de la taille de l'instance) au PAF. La recherche se focalise donc essentiellement sur les méthodes approchées de résolution (abordées à la section 3.8).

3.7 Affectations statique, dynamique et hybride

Il existe trois types d'affectation de fréquences : la statique, la dynamique et l'hybride. L'affectation **statique** (ou fixe) consiste à définir, a priori, l'affectation des fréquences. Le cas **dynamique** (ou *en-ligne*) consiste a affecter progressivement, en temps réel, les fréquences lors de l'arrivée de nouvelles connexions, en veillant à satisfaire les contraintes d'interférence (Audhya et al., 2011; Ghrissi, 2012). L'approche **hybride** (entre le *statique* et le *dynamique*), elle, réserve certaines fréquences statiquement et alloue dynamiquement le reste, à la demande.

Les approches dynamiques et hybrides suggèrent l'existence d'un trafic de signalisation et un accès centralisé au spectre permettant d'éventuelles altérations des affectations existantes (Mahmoudi, 2011). Elles peuvent causer une réaffectation de fréquences aux liens déjà établis, détériorant la performance du réseau vu le temps de convergence nécessaire après l'émission des paquets de contrôle (Yu et al., 2013).

Avec l'augmentation incessante du nombre d'abonnés (et donc potentiellement de requêtes de fréquences), il est primordial d'avoir des affectations au plus proches de l'optimal ainsi qu'un spectre utilisé, au mieux. Or, on peut se permettre des temps de calcul beaucoup plus grand avec l'approche statique que pour les approches dynamique et hybride. Cela permet ainsi de gérer des montées en charge plus importantes (Jin et al., 2001) au vu des solutions (affectation) qui sont de meilleure qualité. De plus, du fait que la grande majorité des travaux récents parlant du PAF (et auxquels nous allons comparer notre travail (en 6.4)) ne traitent que l'aspect statique, nous avons donc décidé de nous limiter à cette approche.

3.8 Méthodes de résolution existantes

Nous présentons ici différentes approches proposées dans la littérature pour la résolution du PAF.

3.8.1 Méthodes exactes

Les méthodes exactes permettent de trouver la solution optimale d'un problème. Compte tenu de la complexité du problème d'affectation de fréquence, ces méthodes ne sont utilisées que lorsque la taille de l'instance à résoudre est petite. Cependant, ces méthodes sont, quelquefois, utilisées pour des instances de taille relativement large mais avec une limite de temps (Dupont et al., 2008). Pour plus de détails concernant les idées d'amélioration de ces méthodes, le lecteur peut se se référer à (Aardal et al., 2007).

3.8.2 Heuristiques spécifiques

Ces méthodes utilisent les spécificités d'un problème afin de trouver une solution approchée en un temps très court. Elles travaillent, soit directement à partir de la formulation du problème, soit à partir d'une modélisation à base de théorie des graphes. Parmi les travaux récents traitant des heuristiques spécifiques à l'affectation de fréquence, on peut citer (Shukla et al., 2009), (Suliman et al., 2013), (Yu et al., 2013) et (Xu et Sakho, 2015).

3.8.3 Méta-heuristiques

Les méta-heuristiques sont des algorithmes généralistes exploitables pour de multiples problèmes. Nous en faisons ici un petit tour d'horizon non exhaustif.

Il y a les algorithmes génétiques (Cotta et Troya, 2001; Abdessemed, 2011) −le point suivant leur est entièrement dédié−, ou d'autres méta-heuristiques évolutionnaires appelées *algorithmes à évolution différentielle* (Kumar Singh et al., 2012). Il y a aussi les méthodes de recherche locale et surtout leurs dérivés, comme la recherche tabou (Kamal et al., 2012; Hadji et Babes, 2013) et le recuit simulé (Wang et al., 2011), les colonies de fourmis (Chouiref et Kaid, 2012), d'abeilles (Mezhoudi, 2010), ou d'autres essaims particulaires (Chakraborty et al., 2008; Zhang et Chen, 2010) ou encore les systèmes immunitaires artificiels (Mahmoudi, 2011; Hadji et al., 2014; Suliman et al., 2014).

D'autres travaux tentent d'améliorer la qualité des solutions en hybridant des méta-heuristiques entre elles (Ghrissi, 2012) ou avec des méthodes issues du *data-mining* (à l'instar de la classification de variables (Kaci et Ait El Hara, 2011) ou du *clustering* (Benyagoub et Chekal Affari, 2013)).

3.9 Les algorithmes génétiques pour le problème d'affectation de fréquences

Plusieurs travaux ont utilisé les algorithmes génétiques pour résoudre le problème d'affectation de fréquences. Cuppini (1994); Ngo et Li (1998); Smith (1998); Chakraborty et Chakraborty (1999); Jin et al. (2001); Acan et al. (2003); Pinagapany et Kulkarni (2008) ont représenté le chromosome comme une matrice (de bits) de dimensions : *Nombre d'émetteurs* x *Nombre de canaux de fréquences*. Chaque ligne de la matrice représente un émetteur et chacune de ses colonnes contient un bit indiquant si le canal de fréquences correspondant est alloué ou non à l'émetteur. Dorne (1995); Alliot et al. (1996); Lai et Coghill (1996); Hurley et al. (1997); Crisan et Miihlenbein (1998); Alabau et al. (2002) ont, eux, employé un codage plus compacte : Le chromosome est segmenté en N segments correspondant chacun à un émetteur. Chaque segment est une liste de taille D_i (demande de l'émetteur i) contenant les canaux fréquentiels alloués à l'émetteur en question.

Le problème avec ces approches est qu'elles permettent d'encoder des solutions non réalisables (qui violent les contraintes de séparation électromagnétique) : Le croisement de deux chromosomes réalisables peut, en effet, générer d'autres qui ne le sont pas. Coello (1999) résume les différentes techniques employées avec les algorithmes génétiques afin de faciliter l'exploration des espaces de recherches avec contraintes :

- **L'emploi de fonctions de pénalité** : Dans cette approche, la fonction objectif est modifiée afin qu'elle pénalise les solutions qui violent les contraintes ;
- **L'utilisation de représentations et d'opérateurs maintenant la faisabilité** : Parmi les approches qui entrent sous cette catégorie, il y a *l'approche de réparation* qui utilise des méthodes de recherche locale pour réparer les solutions violant les contraintes, et *l'approche de décodeurs* qui élimine totalement la région infaisable de l'espace de recherche par construction (approche décrite en détail plus loin).
- **La séparation entre les contraintes et les objectifs** : L'ajout de la minimisation des contraintes violées comme un ou plusieurs objectifs additionnels.

D'après Cotta et Troya (2001), plusieurs travaux traitant le problème d'affectation, dont (Valenzuela et al., 1998; Beckmann et Killat, 1999; Matsui et Tokoro, 2000; Valenzuela, 2001; Matsui et al., 2002, 2003, 2005), ont généré de très bons résultats en employant l'approche avec décodeurs. Dans cette approche, le chromosome encode non pas une solution du problème mais des instructions pour construire une solution. L'algorithme qui décode le chromosome construit la solution en s'assurant de ne pas violer les contraintes ce qui élimine totalement la région non réalisable de l'espace de recherche.

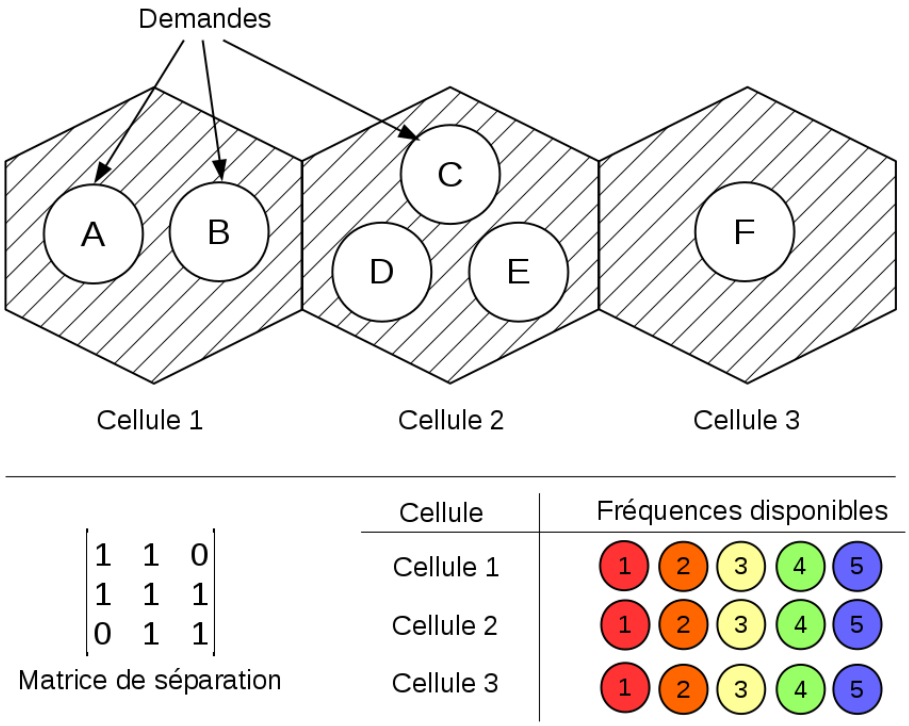


FIGURE 3.2 – Exemple de réseau cellulaire à 3 cellules. La $1^{ère}$ cellule soumet 2 demandes de bandes de fréquences. Ces demandes sont notées A et B. La $2^{ème}$ cellule soumet 3 demandes (C, D et E) et la $3^{ème}$ soumet la demande F. Dans la matrice de séparation M donnée, $M_{i,j}$ représente la séparation minimale requise entre les fréquences affectées aux cellule i et j.

3.9.1 La méthode de Valenzuela et al. (1998)

Valenzuela et al. ont traité la variante de minimisation du *Span* (*MS-FAP*). Le chromosome est représenté par une permutation des demandes de fréquences.

Si on utilise le codage proposé par Valenzuela et al. pour représenter les solutions du problème de la figure 3.2, le chromosome contiendra une permutation des 6 demandes. Un exemple de chromosome est $\{A, F, D, C, B, E\}$.

Pour construire une solution, le chromosome est décodé par un algorithme glouton qui affecte les fréquences aux demandes dans leur ordre d'apparition dans le chromosome, en affectant à chaque étape la première fréquence dont l'ajout ne viole aucune contrainte.

Dans la figure 3.2, le chromosome $\{A, F, D, C, B, E\}$ est décodé comme suit (déroulement schématisé dans la figure 3.3) :

— La fréquence **1** est affectée à la cellule 1 (la soumissionnaire de la demande A). Elle devient interdite dans les cellules 1 et 2 pour satisfaire les contraintes de séparation ;

— La fréquence **1** est affectée à la cellule 3 (la soumissionnaire de la demande F) ;

— La fréquence **2** est affectée à la cellule 2 (la soumissionnaire de la demande D).

Elle devient interdite dans les cellules 1, 2 et 3 pour satisfaire les contraintes de séparation ;

— La fréquence **3** est affectée à la cellule 2 (la soumissionnaire de la demande C) ;

— La fréquence **4** est affectée à la cellule 1 (la soumissionnaire de la demande B) ;

— La fréquence **5** est affectée à la cellule 2 (la soumissionnaire de la demande E).

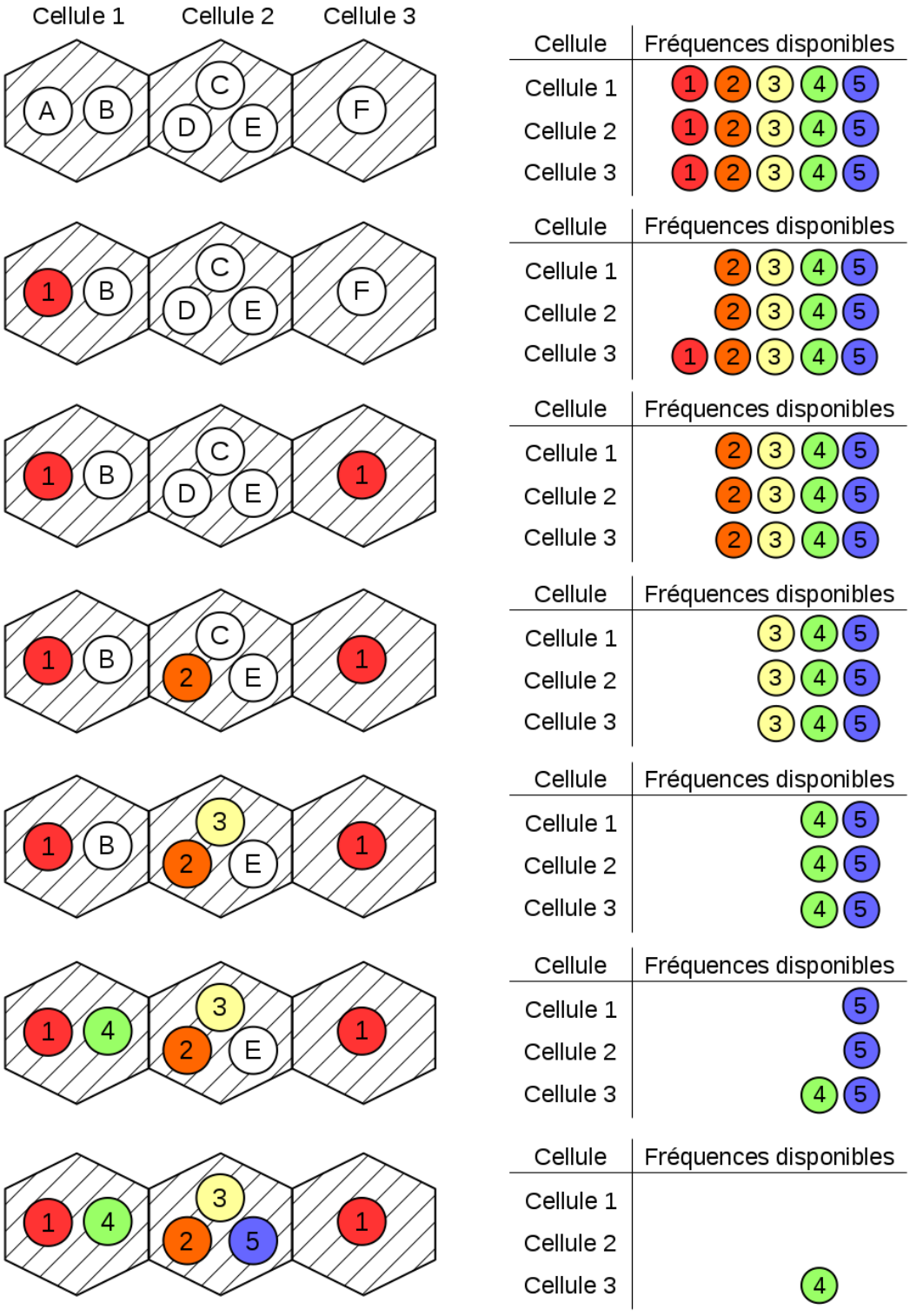


FIGURE 3.3 – Décodage du chromosome $\{A, F, D, C, B, E\}$ (solution du problème de la figure 3.2) par l'algorithme de Valenzuela et al. (1998).

Dans cette approche, les opérateurs génétiques utilisés sont ceux des permutations et plus précisément : *Cycle crossover* et *Order based mutation*. Valenzuela (2001) a étudié la performance des différents opérateurs et a montré que la solution avec *Span* optimal

peut être atteinte avec cette représentation du chromosome et l'approche précédemment décrite.

Afin d'accélérer la convergence de l'algorithme génétique, Valenzuela et al. (1998) ont proposé que la génération de la population initiale utilise l'heuristique *GSD* (*Generalized Saturation Degree*) comme expliqué dans l'algorithme 8.

Algorithme 8 L'algorithme de génération de solutions (permutations) initiales

procédure GSD ()
retourne UNE PERMUTATION
    permutation initiale ← une permutation aléatoire de toutes les demandes ;
    permutation ← une liste vide ;
    fréquences interdites une liste (de taille le nombre de cellules) contenant des listes vides ;
    degrés de saturation ← une liste (de taille le nombre de cellules) contenant des associations (fréquence, séparation) vides ;
    tant que Il y a encore des demandes non traitées **faire**
        $X \leftarrow$ les cellules minimisant $\sum_{f \in fréquences\ interdites(cellule)}(degrés\ de\ saturation[cellule][f])$;
        demande ← la plus prioritaire des demandes des cellules X dans *permutation initiale* ;
        $c \leftarrow$ la cellule soumissionnaire de la *demande* ;
        permutation ← $concaténation(permutation, \{demande\})$;
        Affectation de la $1^{ère}$ fréquence disponible à la *demande* ;
        Mise à jour des *fréquences interdites* ;
        pour chaque cellule c' **faire**
            pour chaque fréquence $f \in fréquences\ interdites(c')$ **faire**
                $degrés\ de\ saturation[c'][f] \leftarrow max(degrés\ de\ saturation[c'][f], séparation(c, c'))$;
            fin pour
        fin pour
    fin tant que
    retourner *permutation*.
fin procédure

3.9.2 La méthode de Matsui et Tokoro (2000)

Matsui et Tokoro ont étendu l'approche de Valenzuela et al. (1998) afin de prendre en compte la propriété d'un graphe (à colorer) stipulant que les éléments d'une clique [12] ne peuvent partager la même couleur. Si k cliques sont identifiées dans le graphe des interférences, le chromosome est segmenté en $k+1$ parties. La $1^{ère}$ partie (ou segment) du chromosome représente une permutation des k cliques et les autres représentent une permutation des demandes appartenant à une clique.

Le chromosome est décodé par un algorithme qui affecte les fréquences aux cliques dans leur ordre de présence dans le 1^{er} segment du chromosome et aux demandes dans

12. En théorie des graphes, une clique est un ensemble de sommets deux-à-deux adjacents.

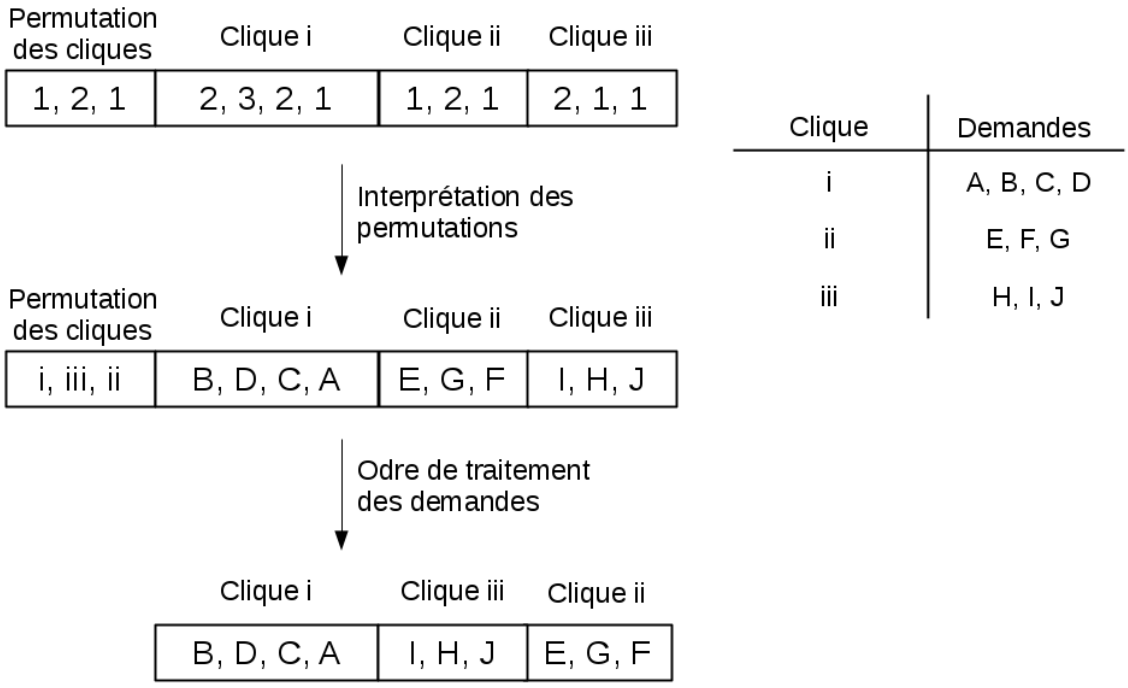


FIGURE 3.4 – Interprétation du chromosome $\{1, 2, 1|2, 3, 2, 1|1, 2, 1|2, 1, 1\}$ en une permutation des demandes $\{A, B, C, D, E, F, G, H, I, J\}$.

leur ordre de présence dans le segment correspondant à leurs cliques, en affectant à chaque étape la première fréquence disponible.

Matsui et Tokoro ont proposé un codage indirect qui représente une permutation de taille N par une liste de nombres entiers de taille N où le i^{eme} élément de la liste admet des valeurs entières dans l'intervalle $[1, (N + 1 - i)]$. La permutation représentée par une telle liste est construite par l'algorithme 9 :

Algorithme 9 Algorithme de Matsui et Tokoro de construction de permutations

```
1: procédure CONSTRUIRE UNE PERMUTATION (CHROMOSOME)
2: retourne UNE PERMUTATION
3:     permutation ← une liste vide ;
4:     tampon ← une liste contenant les N éléments de la permutation ;
5:     pour i allant de 1 à N faire
6:         n ← lire le i^eme élément du chromosome ;
7:         Ajouter le n^eme élément du tampon à la permutation ;
8:         Supprimer le n^eme élément du tampon ;
9:     fin pour
10:    retourner permutation.
11: fin procédure
```

La figure 3.4 donne l'exemple d'un chromosome dans l'encodage de Matsui et Tokoro (2000). Pour décoder la partie $\{2, 1, 1\}$ (La clique *iii* dans l'exemple) en une permutation des lettres H, I et J (Les demandes de la clique *iii*), l'algorithme 9 procède comme suit :

— Initialisation du tampon avec la liste $\{H, I, J\}$ et de la permutation avec la liste vide ;

— Suppression du $2^{ème}$ élément du tampon qui est rajouté à la permutation. Le tampon devient $\{H, J\}$ et la permutation $\{I\}$;

— Suppression du 1^{er} élément du tampon qui est rajouté à la permutation. Le tampon devient $\{J\}$ et la permutation $\{I, H\}$;

— Suppression du 1^{er} élément du tampon qui est rajouté à la permutation. Le tampon devient vide et on obtient comme permutation finale $\{I, H, J\}$.

Ce qui est intéressant dans cet encodage des permutations est le fait que l'utilisation des opérateurs de croisement classiques comme les croisements uniformes ou à N points génèrent toujours des permutations valides. De plus, comme le $i^{ème}$ élément de la liste admet des valeurs entières dans l'intervalle $[1, (N+1-i)]$, la mutation se fait simplement par la génération d'un nombre aléatoire dans cet intervalle.

3.9.3 La méthode basée sur la représentation par clés aléatoires de Matsui et al. (2005)

Matsui et al. ont traité la variante de minimisation du *Span* d'une manière similaire à celle de Valenzuela et al. (1998) mais avec une représentation indirecte des permutations appelée *représentation par clés aléatoires* proposée initialement par Bean (1994) :

Une permutation de N éléments est représentée par une liste de nombres réels dans l'intervalle $[0, 1]$. Pour construire une permutation à partir d'une telle liste, on ordonne les nombres réels (éléments du chromosome) dans l'ordre croissant. Le $i^{ème}$ élément de la permutation est la position originale avant le tri du $i^{ème}$ nombre dans la liste triée.

Une autre interprétation possible de cette représentation est que le $i^{ème}$ nombre réel dans le chromosome représente la priorité inverse de la $i^{ème}$ demande (Plus cette valeur est petite plus la demande est prioritaire). Par exemple, pour décoder le chromosome $\{0.53, 0.71, 0.4\}$ en une permutation des lettres A, B et C, on l'interprète comme suit : 0.53, 0.71 et 0.4 sont les priorités inverses de A, B et C respectivement. D'où la permutation représentée par ce chromosome est $\{D, A, B\}$.

Cet encodage des permutations permet d'utiliser les opérateurs de croisement classiques comme le croisement à N points. La mutation est, elle, effectuée en remplaçant un des nombre par un autre qui est généré aléatoirement, toujours dans l'intervalle $[0, 1]$.

3.9.4 La méthode de Matsui et al. (2003) pour le problème d'affectation de fréquence avec bande fixe

Matsui et al. ont traité le problème d'affectation de fréquences en prenant comme objectifs la minimisation des interférences et la minimisation de blocage. Ils ont représenté

le chromosome comme une séquence d'instructions. Chaque instruction est une paire dont le premier élément représente la cellule à desservir et le deuxième représente une action donnée par un numéro parmi 0, 1 et 2. Les actions peuvent être interprétées comme étant le degré de prise en considération des interférences.

Ainsi, le chromosome $\{(1,0),(2,0),(1,1),(3,2)\}$ est interprété comme suit : Affecter une fréquence à la cellule 1 en utilisant l'action 0, ensuite à la cellule 2 en utilisant l'action 0, puis à la cellule 1 en utilisant l'action 1 et en fin à la cellule 3 en utilisant l'action 2. On dit que chromosome est, en fait, passé à une machine virtuelle qui a opéré l'affectation.

La combinaison des premiers composants de toutes les paires (instructions) forme une permutation de cellules avec répétitions. Les opérateurs de croisement employés sont *Generalized Order Crossover (GOX)*, *Generalized Partially Mapped Crossover (GPMX)*, *Precedence Preservation Crossover (PPX)* et *modified PMX (mPMX)* (Matsui et al., 2003). Le procédé de mutation consiste à échanger les positions de deux paires et à régénérer la partie action par des nombres entiers aléatoires entre 0 et 2.

Conclusion

L'affectation de fréquences est un problème d'actualité ayant un intérêt théorique et pratique certain, que ce soit du point du vue de l'opérateur ou du point de vue de l'abonné (en termes de qualité de service). Il existe, en fait, plusieurs problèmes d'affectation de fréquences qui dépendent du contexte d'application. Au sein d'un même domaine d'application, plusieurs variantes du problèmes sont possibles (et peuvent même être combinées) selon l'objectif poursuivi.

La difficulté intrinsèque du problème fait que la littérature s'oriente, non pas sur la recherche d'algorithmes optimaux mais, sur la recherche d'algorithmes compétitifs. Les algorithmes génétiques figurent ici en bonne place. La modélisation par la théorie des graphes (et notamment par les variantes de la T-coloration) a permis de mieux aborder les spécificités du problème.

Cette modélisation ayant été largement abordée dans la littérature, nous nous proposons d'appliquer une approche différente basée sur l'optimisation multi-objectif et la théorie des jeux. La deuxième partie de ce mémoire est consacrée à la description des approches que nous proposons.

Deuxième partie

Conception et réalisation

Chapitre 4

Conception

Introduction

Nous avons maintenant en main le bagage nécessaire afin d'attaquer notre problématique. Rappelons qu'il s'agit de la résolution du problème d'affectation de fréquence en s'aidant des mécanismes offerts par la théorie des jeux pour l'amélioration des algorithmes de résolution multi-objectif.

Nous allons voulu validé notre travail en utilisant plusieurs benchmarks différents. Nous avons cependant dû adapter notre conception aux spécificités de chacun tout en veillant à séparer la partie qui manipule le problème (les opérateurs génétiques notamment) de l'algorithme de résolution.

Dans ce chapitre, nous allons présenter la conception générale de notre système, expliquer la représentation des instances et des solutions du problème, décrire l'adaptation de l'algorithme génétique que nous avons utilisé, et enfin, expliquer les approches de résolution que nous avons adoptées, adaptées et comment nous les avons améliorées.

4.1 Conception générale du système

Le but de notre travail est la résolution du PAF. Notre système se doit donc de générer, à partir de l'instance donnée, une solution répondant aux objectifs fixés. Nous aurons également besoin de connaître les performances de notre système via différentes métriques qui nous permettront de le comparer aux travaux précédents.

Nous avons décomposé notre système en trois modules principaux :

1. Le module de lecture de l'instance : chargé de la lecture du benchmark et de sa conversion en une structure de données manipulables pour la résolution. En effet, les benchmarks disponibles ont des structures très variées et certains ne supportent pas toutes les variantes du PAF (voir la section 6.1).

2. Le module de résolution : se charge de déterminer l'affectation correspondante à l'instance donnée selon la (ou les) fonction(s) objectif(s) fixée(s).
3. Le module de vérification : fournit des statistiques sur la qualité de service offerte par la solution ainsi que sur la performance du système.

Une interface graphique (qui est indépendante de la résolution) viendra par la suite proposer à l'utilisateur un moyen simple de choisir l'instance, les objectifs et de suivre la résolution.

4.2 Représentation de l'instance et de la solution du problème

Nous avons essayé de généraliser notre conception pour pouvoir l'appliquer à autant de benchmarks que possible. Notre idée était de convertir toutes les instances en un seul format que notre algorithme manipule. Cependant, les benchmarks que nous avons considérés (*Philadelphia* et *COST259*) sont très différents. Ainsi nous avons préféré les traiter séparément afin de tenir compte de leurs spécificités.

Les instances des benchmarks *Philadelphia* et *COST259* partagent, dans notre système, un nombre de structures de données :

— Un vecteur de demande donnant pour chaque émetteur le nombre de fréquences demandées ;

— Une matrice carrée (de taille du nombre d'émetteurs) et donnant la séparation fréquentielle minimale entre les fréquences affectées aux émetteurs. On retrouve sur la diagonale, la contrainte co-cellule et ailleurs la contrainte co-site et des cellules-adjacentes.

Les instances de *COST259* ont en outre :

— La liste des fréquences utilisables pour le réseau ;

— Pour chaque émetteur, la liste des fréquences localement bloquées ;

— Deux matrices carrées donnant la pénalité des interférences co-canal et de canal-adjacent ;

— Une matrice de *Hand-over* précisant pour chaque paire de cellules si il peut y exister un transfert intercellulaire de la première à la seconde, et une matrice spécifiant la séparation fréquentielle minimale requise en cas d'un tel transfert.

— Une différenciation entre les canaux de communication purs (appelés *TCH* [1]) et les canaux faisant également office de canaux de contrôle (appelés *BCCH* [2]. Le canal *BCCH* est le premier canal à être affecté à un site.

1. *Traffic CHannel*
2. *Broadcasting Control CHannel*

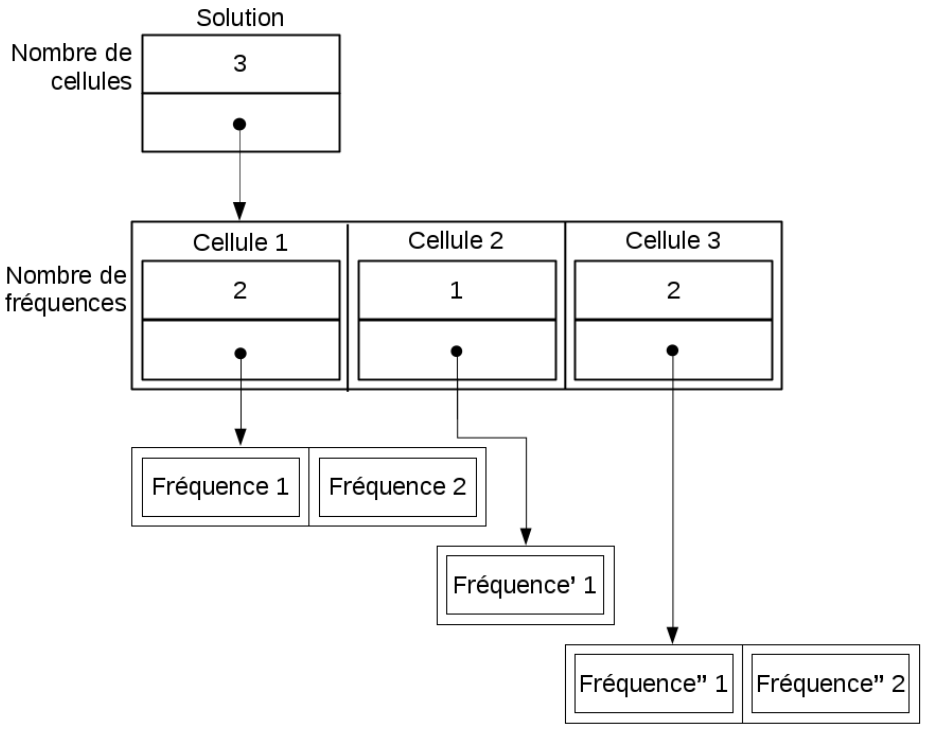


FIGURE 4.1 – Illustration du codage des solutions.

Quant au codage des solutions (affectations), nous avons choisi comme représentation une liste donnant pour chaque cellule la liste des fréquences qui lui sont affectées (Voir la figure 4.1).

4.3 Les fonctions objectifs

Étant donné une instance du problème et une solution, toutes les fonctions objectifs (mentionnées à la section 3.5) sont simplement formulables. Les objectifs que nous avons traités sont la minimisation du *Span* et du *packing* pour les instances du benchmark *Philadelphia*, et la minimisation du blocage et des interférences pour les instances *COST259*.

Les algorithme 10, 11, 12 et 13 décrivent le calcul du *Span*, du *packing*, du blocage et du prix des interférences respectivement et en utilisant, quand nécessaire, les notations définies à la section 3.3.

4.4 Notre adaptation de l'algorithme génétique

La représentation des solutions décrites dans la section précédente peut poser problèmes si elle est utilisée en tant qu'un codage chromosomique, comme nous l'avons mentionné en 3.9. Nous avons donc opté pour l'utilisation de la technique du décodeur en essayant plusieurs codages génétiques.

Algorithme 10 L'algorithme qui calcule le *Span*.

```
1: procédure Calculer le Span(affectation, instance)
2: retourne entier
3:     Min ← +∞
4:     Max ← −∞
5:     pour chaque cellule dans affectation faire
6:         pour chaque fréquence dans affectation[cellule] faire
7:             Min ← minimum(Min, fréquence)
8:             Max ← maximum(Max, fréquence)
9:         fin pour
10:    fin pour
11:    retourner Max − Min
12: fin procédure
```

Algorithme 11 L'algorithme qui calcule le *Packing*.

```
1: procédure Calculer le packing(affectation, instance)
2: retourne entier
3:     C ← fonction qui retourne la séparation co-canal d'une fréquence
4:     packing ← 0
5:     pour chaque fréquence utilisée f faire
6:         pour chaque paire d'émetteurs (e_i, e_j) utilisant f faire
7:             packing ← packing + D(e_i, e_j) − C(f)
8:         fin pour
9:     fin pour
10:    retourner packing
11: fin procédure
```

Algorithme 12 L'algorithme qui calcule le blocage.

```
1: procédure Calculer le blocage(affectation, instance)
2: retourne entier
3:     demandes ← instance[vecteur des demandes]
4:     accumulateur ← 0
5:     pour chaque cellule dans affectation faire
6:         fréquences ← affectation[cellule]
7:         accumulateur ← accumulateur + demandes[cellule] − taille(fréquences)
8:     fin pour
9:     retourner accumulateur
10: fin procédure
```

Algorithme 13 L'algorithme qui calcule le prix des interférences.

```
1: procédure CALCULER LE PRIX DES INTERFÉRENCES(AFFECTATION, INSTANCE)
2: retourne NOMBRE RÉEL
3:     M1 ← instance[matrice des interférences co-canal]
4:     M2 ← instance[matrice des interférences des canaux adjacents]
5:     interférence ← 0
6:     pour chaque cellule c1 dans affectation faire
7:         pour chaque fréquence f1 dans affectation[c1] faire
8:             pour chaque cellule c2 dans affectation faire
9:                 pour chaque fréquence f2 dans affectation[c2] faire
10:                si c1 ≠ c2 et |f1 − f2| = 0 alors
11:                    interférence ← interférence + M1[c1][c2]
12:                fin si
13:                si |f1 − f2| = 1 alors
14:                    interférence ← interférence + M2[c1][c2]
15:                fin si
16:            fin pour
17:        fin pour
18:    fin pour
19:    fin pour
20:    retourner interférence / 2
21: fin procédure
```

Pour *Philadelphia*, nous avons choisi les deux codages génétiques suivant :

— Le chromosome contient une permutation des demandes comme dans (Valenzuela et al., 1998) (décrit en 3.9.1). Pour obtenir la solution représentée par le chromosome, celui-ci est décodé par l'algorithme 14. Les opérateurs de croisement et de mutation sont : *Cycle crossover* et *Position Based Mutation*.

— Le chromosome contient une permutation des demandes codée par la représentation de clés aléatoires comme dans (Matsui et al., 2005). Pour obtenir une solution, le chromosome est transformé en une permutation (comme décrit dans 3.9.3) et est ensuite décodé par l'algorithme 14. Les opérateurs génétiques utilisés sont : *Two Points Crossover* et la mutation par remplacement avec un nombre aléatoire entre 0 et 1 comme dans (Matsui et al., 2005).

Ces codages sont adaptés aux benchmarks *Philadelphia* car ils sont capables d'encoder la solution du *Span* optimal (Valenzuela, 2001). En d'autres termes, il existe un ordre de demandes tel que si on affecte la première fréquence disponible à chaque demande, on arrivera à une solution de *Span* optimal. Le rôle de l'algorithme génétique est donc de découvrir cet ordre. Pour accélérer la convergence de l'algorithme, les solutions initiales sont générées en utilisant l'heuristique *GSD* décrit par l'algorithme 8.

Quant aux benchmarks *COST259*, l'utilisation des codages et du décodeur précédents ne nous donnent aucune garantie vis à vis la qualité des solutions (Que ce soit en terme

Algorithme 14 L'algorithme de décodage du chromosome en une affectation.

```
1: procédure DÉCODER (CHROMOSOME) retourne UNE AFFECTATION
2:     affectation ← une liste de listes vides (une par cellule)
3:     pour i allant de 1 à demande totale faire
4:         demande ← Chromosome[i]
5:         cellule ← la cellule soumissionnaire de la demande
6:         affectation [cellule] ← la première fréquence dont l'ajout ne viole par de
   contraintes.
7:     fin pour
8:     retourner affectation
9: fin procédure
```

de blocage ou de prix des interférences). On remarque que, si elle existe, toute solution réalisable (ne violant aucune contrainte de séparation électromagnétique) et satisfaisant toutes les demandes, peut être obtenue en servant les demandes dans n'importe quel ordre ; il suffit d'affecter la bonne fréquence à chaque étape. Nous avons donc conçu une représentation génétique qui encode les informations : 1) l'ordre de prise en charge des demandes et 2) quelle fréquence à affecter. Le chromosome contient :

— Une liste P contenant D nombres réels entre 0 et 1 (D étant le nombre total des demandes) où la valeur $P[i]$ donne la priorité de la $i^{ème}$ demande (Plus cette valeur est petite plus prioritaire est la demande),

— Une liste F contenant D autres nombres réels entre 0 et 1 où la valeur $F[i]$ spécifie quelle fréquence affecter à la $i^{ème}$ demande : Si on trouve N fréquences disponibles lors du service de la $i^{ème}$ demande, on affecte la $[N \times F[i]+1]^{ème}$ fréquence. Exemple : Si on trouve 10 fréquences disponibles lors du service de la $1^{ère}$ demande et que $F[1] = 0.45$, on affecte la $5^{ème}$ fréquence car $[10 \times 0.45 + 1] = [5.5] = 5$.

Ces deux listes sont entrelacées (croisées) dans le chromosome élément par élément. Le décodage du chromosome en une affectation est fait par l'algorithme 15 qui est basé sur l'heuristique (Haberland, 1996) comme décrit dans (Borndörfer et al., 1998). Le décodeur servit les demandes dans l'ordre défini dans l'heuristique (Haberland, 1996) en se rapportant aux priorités (la liste P) en cas d'indifférence. Mais, au lieu d'affecter la $1^{ère}$ fréquence disponible à chaque étape on affecte celle spécifiée par la liste F. Les figures 4.2 et 4.3 illustrent le déroulement du processus de décodage d'un chromosome (solution du problème de la figure 3.2) par l'algorithme 15.

Les opérateurs génétiques employés pour ce codage génétique sont *Two Points Crossover* et la mutation par remplacement avec un nombre aléatoire entre 0 et 1 (comme dans (Matsui et al., 2005)).

Algorithme 15 L'algorithme de décodage du chromosome en une affectation.

1: **procédure** DÉCODER (CHROMOSOME) **retourne** UNE AFFECTATION
2: $P \leftarrow$ Les éléments du chromosome de rangs impairs
3: $F \leftarrow$ Les éléments du chromosome de rangs pairs
4: **pour** chaque cellule c **faire**
5: Fréquences disponibles[c] $\leftarrow$ Tout le spectre sauf les fréquences localement bloquées dans c.
6: Degrés d'espacement[c] $\leftarrow 0$
7: **pour** chaque cellule c' **faire**
8: **pour** chaque demande de c' **faire**
9: Degrés d'espacement[c] $\leftarrow$ Degrés d'espacement[c] + $séparation(c, c')$
10: **fin pour**
11: **fin pour**
12: **fin pour**
13: affectation $\leftarrow$ une liste de listes vides (une par cellule)
14: **tant que** Il y a encore des cellules avec des demandes non satisfaites et il y a des fréquences disponibles **faire**
15: $X \leftarrow$ les cellules ayant des demandes non encore satisfaites
16: $X \leftarrow$ les cellules de X ayant le nombre minimal de fréquences permises
17: $X \leftarrow$ les cellules de X de degrés d'espacement maximal
18: *demande* $\leftarrow$ Sélectionner une demandes depuis les cellules X qui est la plus prioritaire (selon P)
19: $c \leftarrow$ la cellule soumissionnaire de la *demande*
20: $N \leftarrow$ le nombre de fréquences disponibles
21: *indice* $\leftarrow$ la valeur entière de $F[demande] \times N + 1$
22: affectation [*cellule*] $\leftarrow$ $fréquences\,disponibles[indice]$
23: **pour** chaque cellule c' **faire**
24: Mise à jour de $fréquences\,disponibles[c']$
25: Degrés d'espacement[c'] $\leftarrow$ Degrés d'espacement[c'] $-$ $séparation(c, c')$
26: **fin pour**
27: **fin tant que**
28: **retourner** affectation
29: **fin procédure**

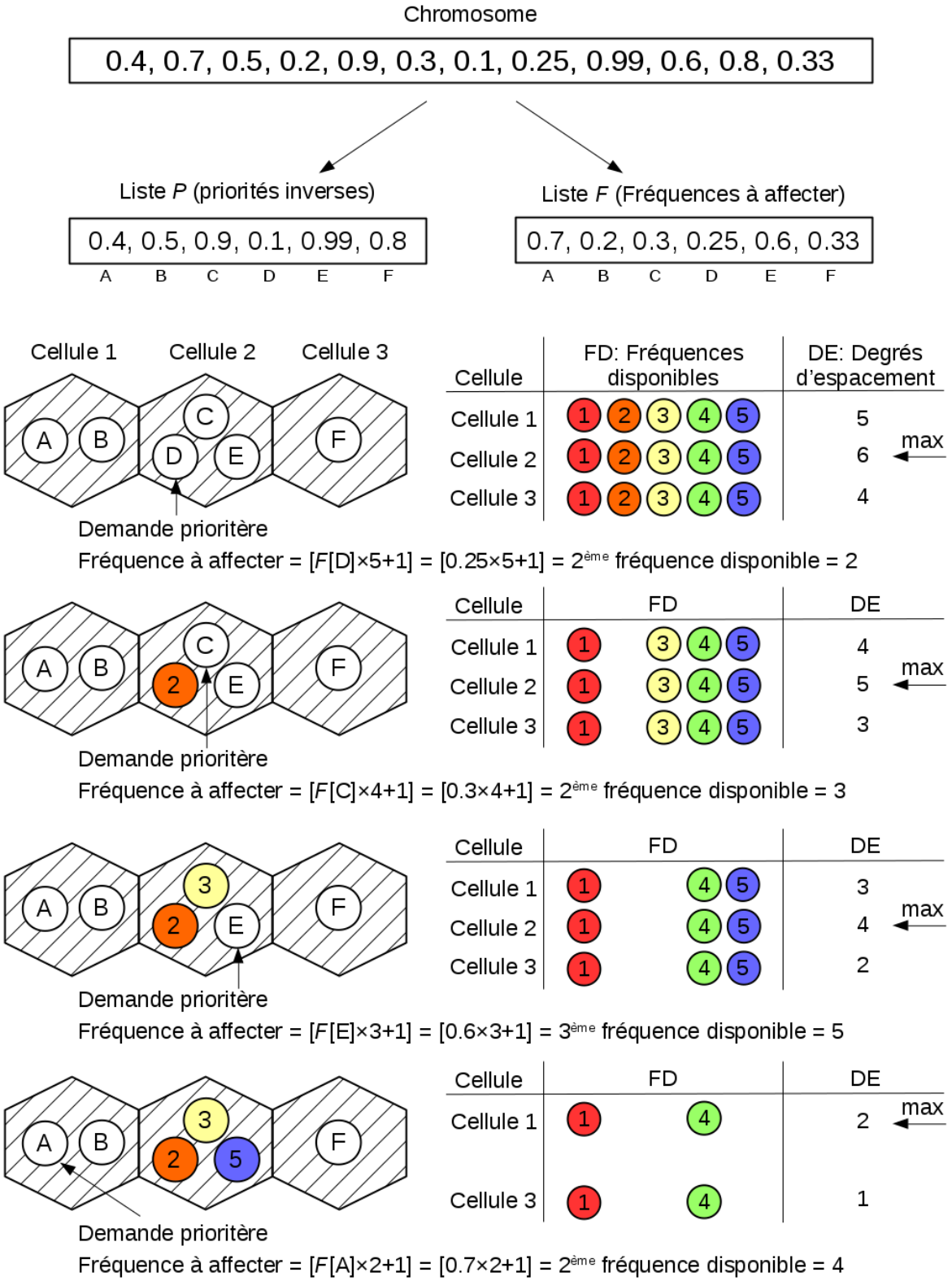


FIGURE 4.2 – Partie 1 du décodage du chromosome $\{0.4, 0.7, 0.5, 0.2, 0.9, 0.3, 0.1, 0.25, 0.99, 0.6, 0.8, 0.33\}$ (solution du problème de la figure 3.2) par l'algorithme 15.

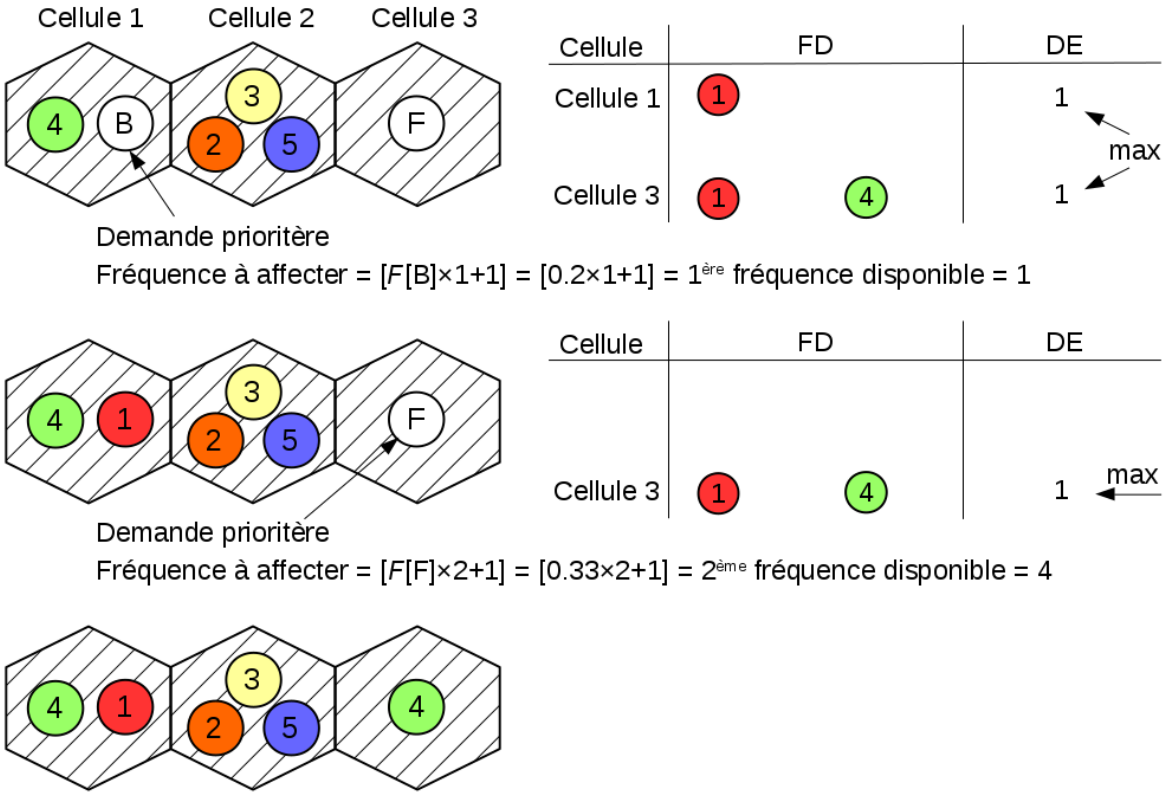


FIGURE 4.3 – Partie 2 du décodage du chromosome $\{0.4, 0.7, 0.5, 0.2, 0.9, 0.3, 0.1, 0.25, 0.99, 0.6, 0.8, 0.33\}$ (solution du problème de la figure 3.2) par l'algorithme 15.

4.5 Approches de résolution adoptées

Nous avons opté pour trois approches de résolutions inspirées de la théorie des jeux : (1) Une approche de solution par équilibre de Nash (*Nash GA*), (2) une approche hybridée combinant équilibre de Nash et optimalité de Pareto (*Nash-Pareto GA*) et (3) une approche co-évolutionnaire (*GCEA*). L'idée est de comparer ces approches aux algorithmes de recherche de la frontière de Pareto (tels *NSGA-II* et *SPEA2*) et d'évaluer l'apport de la théorie des jeux.

4.5.1 Approche basée sur l'équilibre de Nash (Sefrioui et Perlaux, 2000)

Dans cette approche dites *Nash GA*, le processus d'optimisation est modélisé en un jeu à 2 **joueurs** : J_1 et J_2 cherchant à optimiser les fonctions objectifs f_1 et f_2 respectivement. Le chromosome est divisé en 2 parties : La partie 1 représente une **stratégie** du joueur J_1 et la partie 2 représente une stratégie du joueur J_2.

On définit pour chaque joueur i une population de stratégies $Population_i$ et une variable $Meilleure_i$ qui tient sa meilleure stratégie (c.à.d : celle dans $Population_i$ qui

minimise f_i le plus). Le joueur J_1 modifie la valeur de la partie 1 en fixant la partie 2 à la $Meilleure_2$ et vise versa pour le deuxième joueur. Le **profit** tiré par chaque stratégie du joueur i est calculé en évaluant sa concaténation avec la meilleure stratégie de l'autre joueur par la fonction f_i. Ainsi, si $Meilleure_2 = [2, 5, 6, 3, 4, 1]$ alors tous les individus de la $population_1$ seront de la forme $[x, x, x, 3, 4, 1]$ à l'itération suivante (la première partie de solutions sera complétée aléatoirement).

Chaque joueur évolue sa population en exécutant un algorithme génétique et en utilisant les profits comme évaluations de fitness. Les joueurs s'échangent leurs meilleures stratégies à la fin de chaque génération comme expliqué dans l'algorithme 16.

L'algorithme *Nash GA* (comme expliqué dans (Sefrioui et Perlaux, 2000)) ne garanti pas que les solutions non dominées trouvées persistent de génération en génération. Par conséquent, nous avons décidé, dans notre implémentation, d'enregistrer les solutions non dominées dans une archive mise à jour à chaque génération en y ajoutant les solutions nouvellement générées et en supprimant les solutions dominées. Si, après l'ajout de nouvelles solutions, la taille de l'archive devient supérieure au double de la taille d'une population, on exécute la procédure de troncature de *SPEA2* aussi décrite dans (Zitzler et al., 2001). La procédure de mise à jour de l'archive est donnée par l'algorithme 17.

Algorithme 16 L'algorithme *Nash GA* pour 2 objectifs.

procédure L'ALGORITHME *Nash GA*()
  $Archive \leftarrow \emptyset$
  $Meilleure_1 \leftarrow$ une valeur aléatoire ;
  $Meilleure_2 \leftarrow$ une valeur aléatoire ;
  $Population_1 \leftarrow$ un ensemble de valeurs aléatoires ;
  $Population_2 \leftarrow$ un ensemble de valeurs aléatoires ;
  Évaluer $Population(1)$;
  Évaluer $Population(2)$;
  $Meilleure_1 \leftarrow \underset{S \in Population_1}{argmin} \; Fitness(S)$
  $Meilleure_2 \leftarrow \underset{S \in Population_2}{argmin} \; Fitness(S)$
  tant que La condition d'arrêt n'est pas satisfaite **faire**
    pour i allant de 1 à 2 **faire**
      $Sélection(Population_i)$;
      $Croisement(Population_i)$;
      $Mutation(Population_i)$;
      Évaluer $Population(i)$;
      $Tronquer(Population_i, taille\, population)$;
    fin pour
    $Meilleure_1 \leftarrow \underset{S \in Population_1}{argmin} \; Fitness(S)$
    $Meilleure_2 \leftarrow \underset{S \in Population_2}{argmin} \; Fitness(S)$
    pour i allant de 1 à 2 **faire**
      Évaluer $Population(i)$;
    fin pour
    $Nouvelles\, solutions \leftarrow \emptyset$;
    pour i allant de 1 à 2 **faire**
      pour chaque stratégie $S \in Population_i$ **faire**
        $Nouvelles\, solutions \leftarrow Nouvelles\, solutions \cup \{concaténation(S, Meilleur_{3-i})\}$
      fin pour
    fin pour
    $Mise\,à\,jour(Archive, Nouvelles\, solutions)$;
  fin tant que
  retourner *Archive*
fin procédure
procédure ÉVALUER POPULATION(i)
  pour chaque stratégie $S \in Population_i$ **faire**
    $issue \leftarrow concaténation(S, Meilleur_{3-i})$;
    $Fitness(S) \leftarrow f_i(issue)$;
  fin pour
fin procédure
procédure TRONQUER ($Population$, N)
  Trier $Population$ selon les valeurs de $Fitness$
  Supprimer les individus au delà des N premiers
fin procédure

Algorithme 17 L'algorithme de mise à jour de l'archive.

procédure MISE À JOUR(ARCHIVE, NOUVELLES SOLUTIONS)
  Archive $\leftarrow$ Archive $\cup$ Nouvelles solutions ;
  Solutions dominées $\leftarrow \emptyset$;
  pour chaque solution $S \in$ Archive **faire**
    pour chaque solution $S' \in$ Archive - $\{S\}$ **faire**
      si S domine S' **alors**
        Solutions dominées $\leftarrow$ Solutions dominées $\cup\{S'\}$;
      fin si
    fin pour
  fin pour
  Archive $\leftarrow$ Archive $-$ Solutions dominées ;
  tant que |Archive| $>$ seuil **faire**
    { Supprimer la solution la plus proche de tout autre solution (dans l'espace des fonctions objectifs). S'il y en a plusieurs, celle ayant la deuxième plus petite distance est choisie et ainsi de suite.}
    $M \leftarrow$ La matrice de distances entre toutes paires de solutions dans Archive ;
    $X \leftarrow \{i \in [1,|A|]; \exists j \in [1,|A|] - \{i\}, M_{i,j} = \min\limits_{\substack{k\in[1,|A|] \\ l\in[1,|A|]-\{k\}}} M_{k,l}\}$;
    tant que |X| $>$ 1 **faire**
      pour chaque $i \in X$ **faire**
        $j \leftarrow \operatorname{argmin}\limits_{k\in[1,|A|]-\{i\}} M_{i,k}$;
        $M_{i,j} \leftarrow \infty$;
      fin pour
      $X \leftarrow \{i \in X; \exists j \in [1,|A|] - \{i\}, M_{i,j} = \min\limits_{\substack{k\in[1,|A|] \\ l\in[1,|A|]-\{k\}}} M_{k,l}\}$;
    fin tant que
    $S \leftarrow$ Archive[le seul élément dans X] ;
    Archive $\leftarrow$ Archive $-\{S\}$;
  fin tant que
fin procédure

4.5.2 Approche basée sur une hybridation Nash-Pareto (Lee et al., 2010)

Leur hybridation est décrite par l'algorithme 18 dans le cas de deux objectifs (on a besoin de deux joueurs : un par objectif plus un joueur Pareto). Soit un tableau J indexé de 1 à N représentant les populations des joueurs de Nash, une variable JP représentant la population Pareto et un tableau f de fonctions objectifs. Les fonctions suivantes sont

définies :

- *init_alea*(*taille*, *partie*) génère aléatoirement une population de *taille* individus. Le paramètre facultatif *partie* permet de de générer les individus en complétant la partie donnée ;
- *tri*(*population*, *fonction*) trie la *population* selon la *fonction* spécifiée ;
- *getPart*(*individu*, *numéroPartie*) retourne la partie de *individu* ayant pour indice *numéroPartie* ;
- *remplace_partie*(*individu*, *indice*, *partie*) remplace la *indice*-ème part de *individu* par *partie* ;
- *tronquer*(*population*, *taille*) tronque la *population* à la *taille* spécifiée.

Algorithme 18 L'algorithme d'hybridation Nash-Pareto GA (adapté de l'algorithme de Lee et al. (2010)).

$JP \leftarrow init\_alea(taille\_population)$
$tri(JP, f1)$
pour chaque joueur i de 1 à N **faire**
    $J[i+1] \leftarrow init\_alea(taille\_population, getPart(J[i][0], i+1))$
    $tri(J[i], f[i+1])$
fin pour
tant que condition d'arrêt non satisfaite **faire**
    pour chaque joueur i de 1 à N **faire**
        $selection\_tournoi(J[i])$
        pour chaque *indiv* sélectionné de $J[i]$ **faire**
            pour chaque joueur j de 1 à N autre que i **faire**
                $remplace\_partie(indiv, i, P[j][0])$
            fin pour
        fin pour
        $croisement\_mutation(J[i])$
        $trier(J[i])$
    fin pour
    $selection\_NSGA(JP)$
    $croisement\_mutation(JP)$
    pour chaque joueur i de 1 à N **faire**
        $JP.rajout(Ji[0])$
    fin pour
    pour chaque joueur i de 1 à N **faire**
        pour chaque joueur j de 1 à N autre que i **faire**
            $remplace\_partie(indiv, i, P[j][0])$
        fin pour
    fin pour
    $trier\_NSGA(JP)$
    $tronquer(JP, taille\_population)$
fin tant que
retourner JP

4.5.3 Approche basée sur les jeux co-évolutionnaires (Sim et al., 2004)

Dans cette approche, on crée 2 **joueurs** : J_1 et J_2 cherchant à optimiser les fonctions objectifs f_1 et f_2 respectivement. Chaque joueur i manipule une population $Population_i$ de solutions. Chaque solution x représente une **stratégie** et le **profit** qu'elle génère est calculé en la misant en jeu avec une stratégie x' aléatoirement choisie de la population adversaire : Si $x \in Population_1$ et $x' \in Population_2$ alors $Profit(x) = (f_1(x) - f_2(x'))$ et $Profit(x') = (f_2(x') - f_1(x))$.

Chaque joueur exécute un algorithme génétique pour évoluer sa population en utilisant les profits comme des évaluations de fitness des solutions. Cela est expliqué dans l'algorithme 19.

L'algorithme, appelé *GCEA* (*Game Theory Based Co-evolutionary Algorithm*) ne sauvegarde pas nécessairement les solutions non dominées (tout comme *Nash GA*). Nous avons donc utilisé la même technique d'archivage décrite au point 4.5.1 par l'algorithme 17.

Algorithme 19 L'algorithme *GCEA*

1: **procédure** L'ALGORITHME *GCEA*()
2: $Archive \leftarrow \emptyset$
3: Générer $Population_1$ aléatoirement
4: Générer $Population_2$ aléatoirement
5: **pour** i allant de 1 à $taille\,population$ **faire**
6: $x \leftarrow Population_1[i]$
7: $x' \leftarrow Population_2[i]$
8: $Fitness(x) = (f_1(x) - f_2(x'))$
9: $Fitness(x') = (f_2(x') - f_1(x))$
10: **fin pour**
11: **tant que** La condition d'arrêt n'est pas satisfaite **faire**
12: **pour** i allant de 1 à 2 **faire**
13: $Sélection(Population_i)$;
14: $Croisement(Population_i)$;
15: $Mutation(Population_i)$;
16: **fin pour**
17: Mélanger les individus de $Population_1$
18: Mélanger les individus de $Population_2$
19: **pour** i allant de 1 à $taille\,population$ **faire**
20: $x \leftarrow Population_1[i]$
21: $x' \leftarrow Population_2[i]$
22: $Fitness(x) = (f_1(x) - f_2(x'))$
23: $Fitness(x') = (f_2(x') - f_1(x))$
24: **fin pour**
25: $Tronquer(Population_1, taille\,population)$;
26: $Tronquer(Population_2, taille\,population)$;
27: $Mise\,à\,jour(Archive, Population_1 \cup Population_2)$;
28: **fin tant que**
29: **retourner** *Archive*
30: **fin procédure**
31: **procédure** TRONQUER ($Population$, N)
32: Trier $Population$ selon les valeurs de $Fitness$
33: Supprimer les individus au delà des N premiers
34: **fin procédure**

Conclusion

Dans ce chapitre, nous avons explicité la conception de notre système, les approches que nous avons choisies et les améliorations proposées (notamment le rajout d'une archive pour les deux algorithmes *Nash GA* et *GCEA*). Nous avons décrit la représentation des instances et des solutions du problème ainsi que notre adaptation de l'algorithme génétique. Ainsi, nous avons pu séparer l'algorithme de résolution en lui même, de la partie qui gère les spécificités du problème. Ceci permet de changer aisément de représentation de solution ou d'opérateurs génétiques.

Maintenant que nous avons exhiber les détails de nos algorithmes de résolution, il ne reste plus qu'à les mettre sur machine et à tester leur performances.

Chapitre 5

Implémentation

Introduction

Dans le chapitre précédent, nous avons décrit le fonctionnement des algorithmes de résolution du PAF multi-objectif utilisant la théorie des jeux que nous allons comparer aux algorithmes (*NSGA-II* et *SPEA2*).

Dans ce chapitre, nous allons expliciter les détails de l'implémentation de l'ensemble de ces approches.

5.1 Langages de programmation et environnement d'exécution

Nous avons choisi d'utilisé le langage de programmation **C++**. Il s'agit d'un langage compilé multi-paradigme (notamment procédural et orienté-objet) et multi-plateforme, standardisé par l'ISO [1] dès 1998. Il est réputé pour offrir des structures de données rapides avec une manipulation abstraite ce qui permet d'avoir des codes portables et néanmoins efficaces. De plus, de nombreux livres et tutoriels traitant de ce langage sont disponibles sur le web et maintenus par une communauté active. Cette dernière propose aussi de nombreuses bibliothèques spécialisées.

Nous avons exploité les dernières améliorations du langages et de sa bibliothèque standard étoffées dans les versions modernes appelées C++11 et C++14.

Nous avons travaillé sous environnement Linux avec le compilateur libre *g++* et le profileur *valgrind* à travers l'environnement de développement intégré *QtCreator* qui permet le maniement de ces outils d'une manière graphique très intuitive.

1. *International Organization for Standardization*

5.2 Méthodologie de développpement

Nous avons développé notre système en incrémentale pour la partie codage après la finalisation de la conception globale. Ainsi, nous avons distinguer les incréments suivants :

1. Lecture des benchmarks et stockage des instances ;
2. Développement des opérateurs génétiques et d'un algorithme génétique simple ;
3. Développement des algorithmes *NSGA-II* et *SPEA2* ;
4. Développement de chacune des trois approches adoptées ;
5. Parallélisation et optimisation du code ;
6. Réalisation de l'interface graphique.

Chaque incrément est testé à part avant les tests du chapitre 6.

Nous avons travaillé en mode collaboratif avec gestion de version par le logiciel *git* et hébergement sur la plateforme *bitbucket* [2].

5.3 Réalisation

Nous avons opté pour une implémentation orientée-objet afin de bénéficier des avantages fournis par les mécanismes d'héritage, de redéfinition/spécialisation ainsi que pour avoir une structuration logique du code. Nous décrivons ici les classes et des structures de données les plus importantes de notre implémentation comptant prés de 5000 lignes de code [3].

5.3.1 Lecture de l'instance Philadelphia

Nous avons privilégié l'utilisation d'une structure de données simples étant donné que nous n'avons besoin que de lire l'instance (ce qui, dans ce cas, est très simple).

Champs	Description
`demand`	Vecteur de demande
`total_demand`	Somme des demandes (du vecteur précédent)
`separation_matrix`	La matrice de séparation fréquentielle minimale
`distance_matrix`	La matrice de séparation géographique inter-émetteurs
`Philadelphia_instance()`	Constructeur de la structure à partir d'un flux d'entrée (fichier)

TABLEAU 5.1 – Description de la structure *Philadelphia_instance*

2. Cette même façon de faire nous a servi à rédiger ce document via le logiciel libre de mise en forme de documents LaTeX
3. Code de l'interface graphique non inclus.

5.3.2 Lecture de l'instance COST

Même chose que pour le premier type de benchmark sauf que celui-ci possède plus de contraintes.

TABLEAU 5.2 – Description de la structure *COST259*

Champs	Description
`original_cell_ids`	Mapping entre les indices des sites (commençant à 0) et les indices dans le benchmark
`new_cell_ids`	Mapping inverse du précédent
`minimum_channel`	Canaux limites de la bande utilisable
`maximum_channel`	
`globally_blocked_channels`	Liste des canaux bloqués pour tout le réseau
`locally_blocked_channels`	Liste des canaux bloqués pour chaque émetteur
`nb_cells`	Nombre de cellules
`demand`	Vecteur de demande
`total_demand`	Nombre total de demandes en canaux
`bcch_bcch`	Entiers spécifiants les valeurs des séparations fréquentielles minimales entre canaux selon leur types
`bcch_tch`	
`tch_bcch`	
`tch_tch`	
`handover_matrix`	Matrice de séparation fréquentielle minimale
`separation_matrix`	Matrice de séparation
`interference_matrix`	Matrice spécifiant l'interférence générée lors de l'affectation
`violated_constraints()`	Retourne le nombre de contraintes violées par une affectation
`COST259_instance()`	Constructeur de copie
	Lecture à partir du fichier de benchmark

5.3.3 Décodage et opérations sur le chromosome : la classe Variator

La classe abstraite (ou interface) Variator (dénomination inspirée de la plateforme PISA (Bleuler et al., 2003)) encapsule une entité qui se charge du décodage, de la génération de solutions initiales et d'effectuer les opérations génétiques. Elle permet d'écrire des algorithmes génétiques qui fonctionnent avec plusieurs codages (Comme ceux décrits dans la section 3.9).

Cette classe est ensuite implémentée par les quatre classes : `Philadelphia_variator`,

TABLEAU 5.3 – Description de l'interface *Variator*

Champs	Description
`decode()`	Génére une affectation valide à partir d'une permutation de demandes
`crossover()`	Effectue un crossover entre deux chromosomes si le générateur aléatoire donne un nombre inférieur à la probabilité de crossover
`mutate()`	Effectue une mutation du chromosome passé en paramètre si le générateur aléatoire donné donne un nombre inférieur à la probabilité de muation
`create()`	Créé un chromosome (une permutation) aléatoire
`create_from()`	Modifie une partie définie du chromosome donné

`Philadelphia_random_key_variator`, `COST259_rk_aftt_variator` et `COST259_random_key_variato`
Les deux premiers variateurs sont sémantiquement équivalent : Ils encodent des permutations de demandes de fréquences avec la différence que `Philadelphia_variator` utilise un codage génétique par permutation alors que `Philadelphia_random_key_variator` utilise un codage par clés aléatoire. Quant au variateur `COST259_rk_aftt_variator`, il encode les listes P et F comme éxpliqué dans 4.4. Ainsi, Le role de l'algorithme génétique devient d'explorer l'espace des priorités (P) des demandes de fréquences et l'espace des fréquences à choisir (F). La différence entre ce dernier variateur et `COST259_random_key_variator` est que celui ci n'explore que l'espace de fréquences à choisir (F).

5.3.4 Evolution et visualisation des populations : l'interface GA

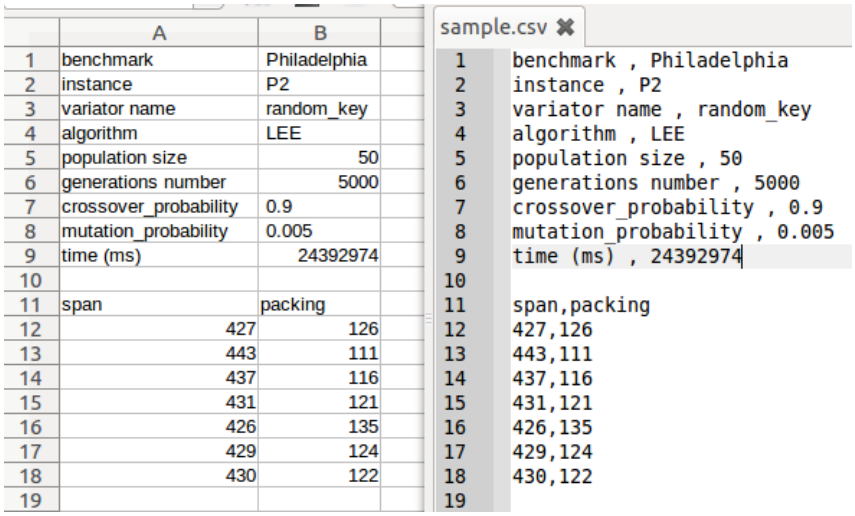
TABLEAU 5.4 – Description de l'interface *GA*

Champs	Description
`evolve()`	Réalisation d'un passage de génération
`log()`	Permer de visualiser l'avancement de l'algorithme
`get_solutions()`	Retourne les solutions non dominées trouvées

Cette classe est implémentée par les différentes classes des algorithmes de résolution : `Single_objective_ga`, `NSGA`, `SPEA`, `Nash_GA`, `GCEA` et `Lee` (hybridation Nash-Pareto).

Le fichier log est structuré de sorte à pourvoir être employer au format *CSV* [4] afin ed pourvoir être facilement être lu à partie d'un éditeur de texte simple ou d'un tableur. La figure 5.1 donne un exemple de fichier log.

4. *Comma-Separated Values* ou valeurs spéarées par des virgules



	A	B
1	benchmark	Philadelphia
2	instance	P2
3	variator name	random_key
4	algorithm	LEE
5	population size	50
6	generations number	5000
7	crossover_probability	0.9
8	mutation_probability	0.005
9	time (ms)	24392974
10		
11	span	packing
12	427	126
13	443	111
14	437	116
15	431	121
16	426	135
17	429	124
18	430	122
19		

FIGURE 5.1 – Exemple de fichier *log* généré donnant les paramètre de l'algorithmes ainsi que les valeurs des fonctions objectifs des solutions trouvées

5.3.5 Programme principal

La structuration du code décrite permet facilement l'ajout, de manière simple, d'autres méthodes de résolution ou d'autres `variators`. Ainsi, le programme principal ne manipule qu'un seul objet `variator` qu'il utilise pour créer l'objet `GA` selon le choix de l'utilisateur. Il ne reste plus qu'à itérer l'appel aux méthodes `evolve` et `log` de l'objet `GA`, puis, une fois le nombre d'itérations requis atteint, à utiliser la méthode `get_solutions` afin d'enregistrer les résultats trouvés.

5.4 Parallélisation et optimisation du code

Afin d'obtenir une amélioration en termes de temps d'exécution, nous avons parallélisé certaines portions de code et plus précisément les boucles d'initialisation des populations et d'évolution des populations. Nous avons déterminé que ces parties étaient paralléliseables et, en effectuant le profilage de code (*profiling*) de notre code séquentiel, que c'était ces parties prenaient le plus de temps.

Nous avons utilisé la bibliothèque **OpenMP** qui offre un ensemble de directives pour l'écriture de programmes parallèles en utilisant des *threads* avec une mémoire partagée. Notre choix a été guidé par la simplicité d'utilisation, les fonctionnalités nombreuses qu'offre cette bibliothèque (la synchronisation ou les sections critiques par exemple) et la communauté nombreuse qui la maintient et la fait évoluer.

Nous avons donc légèrement modifié notre code afin de rendre les itérations des boucles à paralléliser indépendantes entre-elles puis introduit les directives *OpenMP* en suivant la pratique consistant à paralléliser les boucles les plus externes au lieu des plus internes et en déterminant, à l'exécution, le nombre de cœurs de processeurs de la machine et créer

un nombre de *threads* équivalents.

5.5 Interface graphique

L'interface graphique ne représente, certes, pas le cœur de notre travail, ni un composant essentiel, nous avons voulu quand même une interface graphique simple et fonctionnelle afin de permettre à l'utilisateur de manipuler aisément d'ajuster les paramètres, de lancer la résolution ou de consulter le *log* d'exécution. Elle est réalisée sur la plateforme *Qt* disponible en licence libre et qui offre l'avantage d'être multi-plateforme et de bien se marier avec le langage C++.

Conclusion

Nous avons présenté la structuration du code implémentant notre travail. Nous avons fait en sorte de le structurer et d'optimiser son exécution au maximum afin d'obtenir les meilleurs résultats possibles. Ainsi, nous avons séparé les parties chargées de la gestion des spécificités des benchmarks, des algorithmes et nous avons parallélisé l'exécution tout en altérant au minimum la structure du code.

Les résultats de l'exécution font l'objet de la dernière partie de ce mémoire.

Troisième partie

Tests et résultats

Chapitre 6

Tests et résultats

Introduction

Nous avons eu la chance d'avoir accès au cluster *IBNBADIS* [1] du *CERIST* [2] qui offre des machines Intel® Xeon® dotées d'une mémoire vive de 64 *Go* et de 2 processeurs de 8 cœurs chacun, cadencés à 2 GHz. Nous y avons effectué nos tests en *multi-thread* (sur une machine) en employant l'ensembles de cœurs de processeur

Nous avons précédemment cité les deux benchmarks que nous avons utilisés (*Philadelphia* et *COST259*). Nous les décrivons ici. Dans l'étude empirique, nous déterminerons les meilleures valeurs des paramètres pour chaque algorithme puis nous les comparerons entre eux. Nous comparerons aussi nos résultats avec des travaux récents traitant du PAF.

6.1 Benchmarks de test utilisés

De nombreuses instances de test existent dans la littérature pour le PAF. Néanmoins, chaque auteur utilise un ou plusieurs de ces benchmarks. Certains génèrent même des benchmarks aléatoires ou utilisent des instances propriété d'opérateurs, donc impossible à utiliser. Nous avons choisi les benchmarks les plus employés pour nos tests en faisant en sorte que leurs variétés nous permettent d'expérimenter les différentes variantes du PAF, ce qui n'était pas possible avec certains d'entre eux. Voici une brève description de chacun d'eux.

1. dans le cadre du *Réseau Algérien sur le Calcul Intensif et Modélisation (RACIM)* www.rx-racim.cerist.dz/?page_id=26
2. Centre de Recherche sur l'Information Scientifique et Technique

6.1.1 Philadelphia

Ce benchmark[3], très utilisé (voire le plus utilisé), représente une modélisation de la ville américaine de Philadelphie (en Pennsylvanie) divisée en 21 cellules hexagonales avec 9 instances[4] qui diffèrent par le vecteur de demande et les séparations fréquentielles minimales. Cette instance ne peut être utilisée avec la variante MI-FAP puisqu'elle n'indique pas les interférences (ou les pénalités qu'elles induisent) mais uniquement les séparations à respecter.

Ce benchmark est certainement le plus utilisé dans la littérature concernant le PAF. Les solutions optimales sont connues (en termes de span minimum) et permettent de juger plus aisément de la qualité des solutions obtenues. Les instances 2 et 6 sont réputées être les plus difficiles.

6.1.2 COST 259

Les 32 instances de ce benchmark[5] proviennent de plusieurs opérateurs de télécoms. Ces instances sont appelées : Bradford_nt-t-p (15 instances), Bradford-t-p (12 instances), Siemens (4 instances), K et Swisscom.

Ce benchmark se veut être plus proche des problématiques réelles. Il en intègre nombre de spécificités pratiques comme des fréquences bloquées globalement ou localement au niveau des émetteurs, des spectres disjoints de fréquences, la distinction entre le premier canal alloué à un émetteur (gérant à la fois des communications et des données de contrôle) et les autres dédiés aux communications.

6.2 Choix des variateurs

Nous avons développé plusieurs représentations génétiques et décodeurs pour chacun des benchmarks *Philadelphia* et *COST259*. Afin de minimiser le nombres de testes à effectuer dans l'étude empirique et comparative, nous avons choisi un seul variateur par benchmark en lançant une série de tests similaires à ceux qui seront présentés dans la section 6.4.

Il ressort que les meilleurs représentations sont : la représentation génétique *Permutation avec clés aléatoires* pour *Philadelphia* et la représentation génétique par listes P et F (décrites en 4.4) pour le benchmark *COST259*.

3. dont la description totale est disponible ici : `fap.zib.de`
4. Deux instances ont un nombre différent de cellule : 4 et 25
5. également disponible sur le site dédié du *Zuse Institute Berlin (ZIB)*

6.3 Étude empirique

Au lieu de faire varier un à un les paramètres en laissant les autres fixés à une certaine valeur, nous avons décidé (vu les capacités de calcul disponibles) de travailler avec les combinaisons de paramètres afin de trouver un meilleur ajustement de paramètres pour chaque algorithme.

Nous avons, pour chaque algorithme, pour chaque combinaison des paramètres choisis, et pour chacune des deux instances (Nous avons choisi l'instance *P2* comme représentant du benchmark Philadelphia et l'instance *Swisscom* pour *COST259* au vu de leur taille ni trop petite, ni trop grande) effectué 5 exécutions : le but étant de pouvoir distinguer l'aspect aléatoire (présent dans l'algorithme) de la résolution en elle même. Nous avons utilisé la métrique C présentée au point 2.2 pour effectuer la comparaison. Nous obtenons un grand tableau possèdant la même structure que le tableau 2.1. Les résultats étant trop volumineux pour être entièrement décrits ici, nous avons décidé de les donner sous format électronique (disponibles à l'adresse `https://goo.gl/kIxfDJ`) et de donner ici les résultats finaux et une discussion ces derniers.

6.3.1 Méthodologie de l'étude empirique

Nous avons effectué pour chaque paire (algorithme, benchmark) 5 exécutions (générant 5 fronts de Pareto) pour chaque combinaison de paramètres (taille de population, nombre de générations, probabilité de croisement et probabilité de mutation).

Puis, pour chaque paire (algorithme, benchmark) et pour chaque combinaison de paramètres, nous avons choisi le meilleur parmi les 5 fronts $(f_1, .., f_5)$ générés comme suit : Nous avons calculé une matrice M de taille 5×5 où $M_{i,j} = C(f_i, f_j)$. Tout comme $C(f_i, f_j)$ mesure la proportion du front f_j couverte par le front f_i, on cherche le front f_j qui minimise le plus $C(f_i, f_j)$ pour $i \neq j$. Nous avons donc calculé la moyenne des colonnes de la matrice ($Moyenne(M_{*,j}) = Moyenne(*, f_j)$ pour tout j) et choisi le front f_j qui est de moyenne minimale.

Maintenant que nous avons un front de Pareto par paire (algorithme, benchmark) et par combinaison de paramètres, on cherche la combinaison de paramètres optimale pour chaque paire (algorithme, benchmark). Nous avons calculé pour chaque paire (algorithme, benchmark) une matrice M de taille (le nombre de combinaisons de paramètres)^2 avec $M_{i,j} = C(f_i, f_j)$ avec $M_{i,j} = C(f_i, f_j)$. En suite, nous avons choisi le front f_j qui minimise la moyenne de la colonne j ($Moyenne(M_{*,j})$). Les paramètres utilisés pour générer ce front sont pris comme paramètres optimaux de la paire (algorithme, benchmark).

Les résultats de notre étude empirique sont donnés dans les sections ci-après. Nous nous sommes basés, pour les paramètres *probabilité de croisement* et *probabilité de mutation*, sur le fait que la littérature traitant des algorithmes génétiques recommande d'utiliser des

taux proches de 1 pour la première et proches de 0 pour la seconde.

6.3.2 Les algorithmes *NSGA-II* et *SPEA2*

Nous avons testé les combinaisons de paramètres formés des valeurs présentées dans le tableau 6.1.

TABLEAU 6.1 – Valeurs des paramètres pour les algorithmes *NSGA-II* et *SPEA2*

Paramètres	**Valeurs**
Taille de la population	50 ; 100
Nombre de générations	1000 ; 2000 ; 5000
Probabilité de croisement	0.75 ; 0.9 ; 0.95 ; 1
Probabilité de mutation	0.0005 ; 0.005 ; 0.05

Les meilleurs résultats sont donnés dans le tableau 6.2.

TABLEAU 6.2 – Meilleurs paramètres pour les algorithmes *NSGA-II* et *SPEA2*

Paramètres \ Algorithmes	**NSGA-II**		**SPEA2**	
	COST259	Philadelphia	COST259	Philadelphia
Taille de la population	100	100	100	100
Nombre de générations	5000	2000	2000	5000
Probabilité de croisement	1	0.9	0.9	1
Probabilité de mutation	0.005	0.005	0.005	0.0005

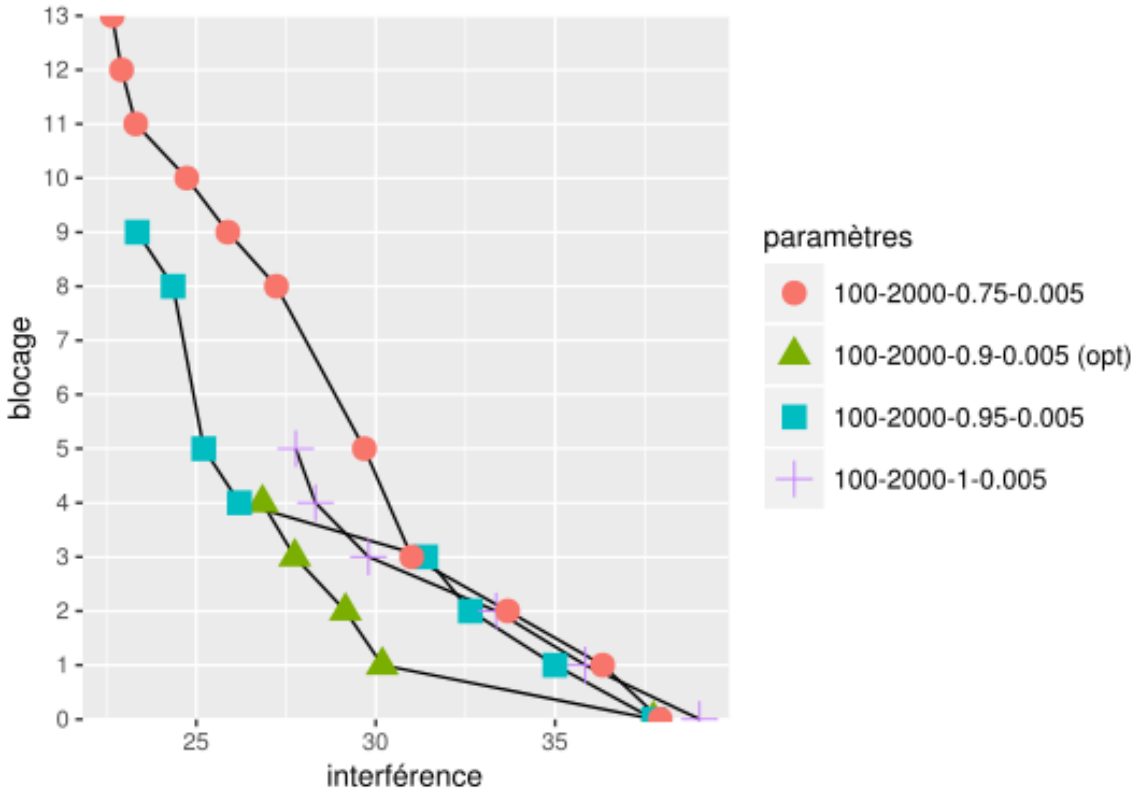


FIGURE 6.1 – Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *SPEA2* sur le benchmark *COST259*

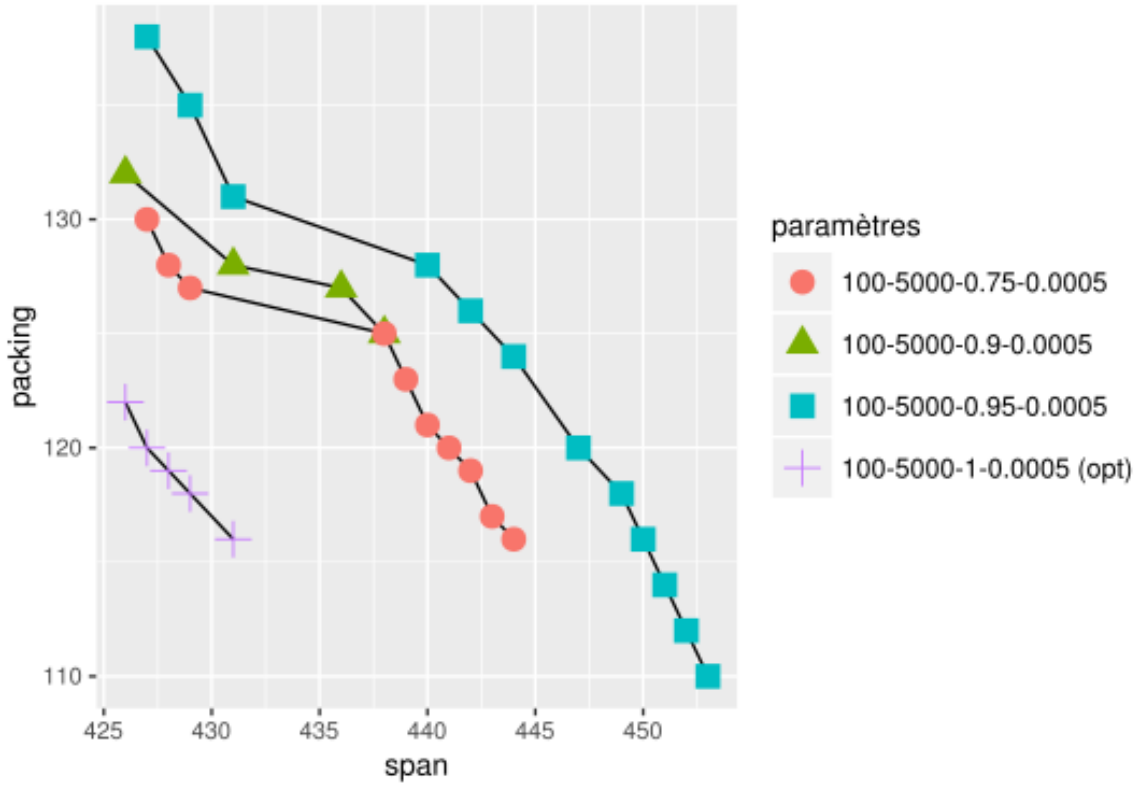


FIGURE 6.2 – Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *SPEA2* sur le benchmark *Philadelphia*

Les figures[6] 6.1 et 6.2 illustrent la comparaison entre la meilleure combinaison de paramètres (opt) de l'algorithme *SPEA2* et les combinaisons où seule la probabilité de croisement a été modifiée. Nous remarquons que la probabilité de croisement optimale varie en fonction du benchmark : Elle est de 0.9 pour *COST259* et de 1 pour *Phildelphia*.

La figure 6.3 compare la meilleure combinaison de paramètre (opt), de l'algorithme *NSGA-II* et le benchmark *COST259*, et les combinaisons où seule la probabilité de mutation a été modifiée. Elle montre bien la difficulté de paramétrage de cette probabilité qui ne doit ni être trop petite et ne pas apporter assez de diversité de solution ni trop grande et ne pas permettre une recherche poussée.

6. Dans toutes les figures de l'étude empirique, la notation suivante est adoptée (taille de population - nombre de générations - probabilité de croisement - probabilité de mutation)

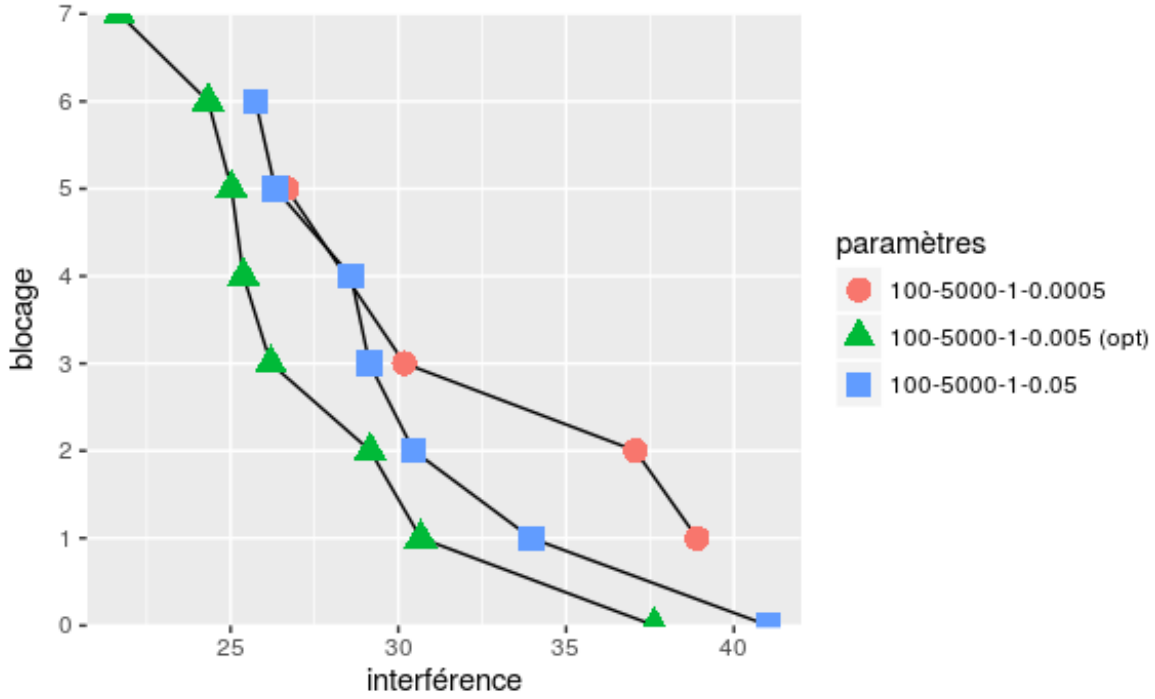


FIGURE 6.3 – Comparaison entre 3 combinaisons de paramètres faisant varier la probabilité de mutation de l'algorithme *NSGA-II* sur le benchmark *COST259*

6.3.3 Les algorithmes *Nash GA* et *Nash-Pareto GA*

Nous avons testé les combinaisons de paramètres formés des valeurs présentées dans le tableau 6.3.

TABLEAU 6.3 – Valeurs des paramètres pour les algorithmes *Nash GA* et *Nash-Pareto GA*

Paramètres	**Valeurs**
Taille de la population	25 ; 50
Nombre de générations	1000 ; 2000 ; 5000
Probabilité de croisement	0.75 ; 0.9 ; 0.95 ; 1
Probabilité de mutation	0.0005 ; 0.005 ; 0.05

Les meilleurs résultats sont donnés dans le tableau 6.4.

TABLEAU 6.4 – Meilleurs paramètres pour les algorithmes *Nash GA* et *Nash-Pareto GA*

Paramètres \ Algorithmes	**Nash-Pareto GA**		**Nash GA**	
	COST259	Philadelphia	COST259	Philadelphia
Taille de la population	50	50	25	50
Nombre de générations	5000	5000	1000	5000
Probabilité de croisement	0.95	0.9	1	1
Probabilité de mutation	0.005	0.005	0.005	0.005

La figure 6.4 montre une comparaison entre les résultats obtenus par la meilleure combinaison de paramètres (opt) de l'algoritme *Nash GA* et des combinaisons (sur le benchmark COST259) où seul la probabilité de mutation a été changée. Notons ici que, une probabilité de croisement fixée à 1 permet nettement d'obtenir de meilleurs résultats qu'avec des valeurs inférieures. Ceci n'est pas le cas pour l'algorithme *Nash Pareto GA* comme l'illustre la figure 6.5 où la probabilité de croisement égale à 1 dégrade la qualité de la frontière.

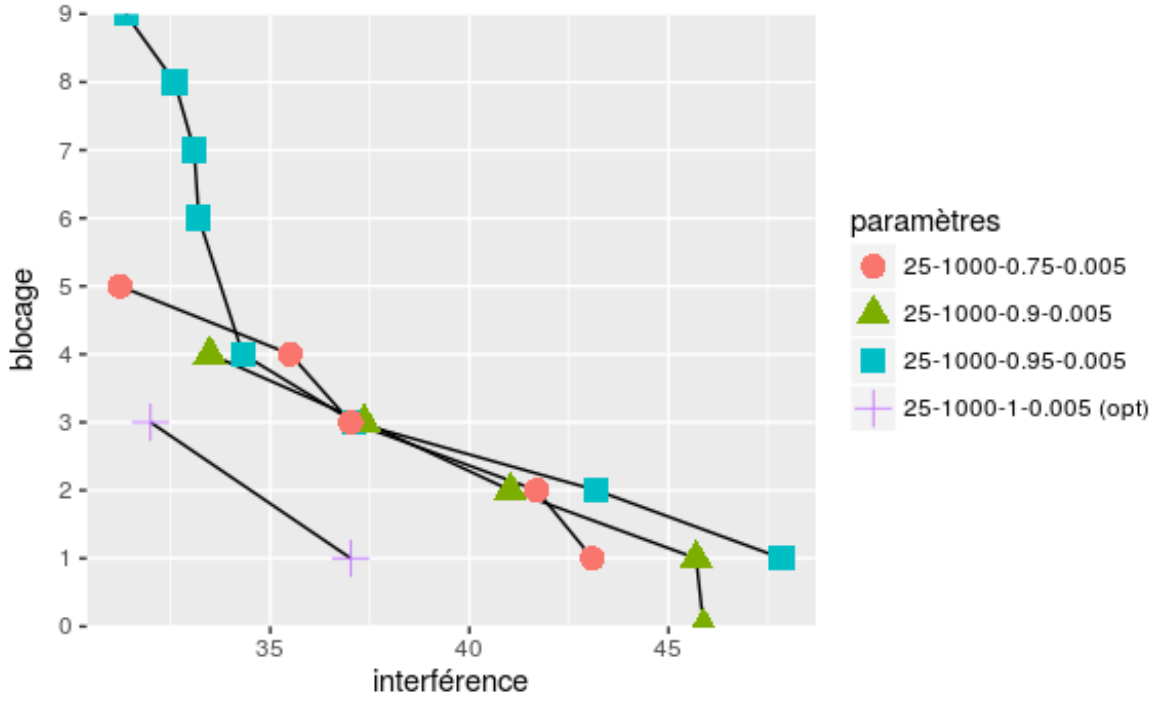


FIGURE 6.4 – Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *Nash GA* sur le benchmark *COST259*

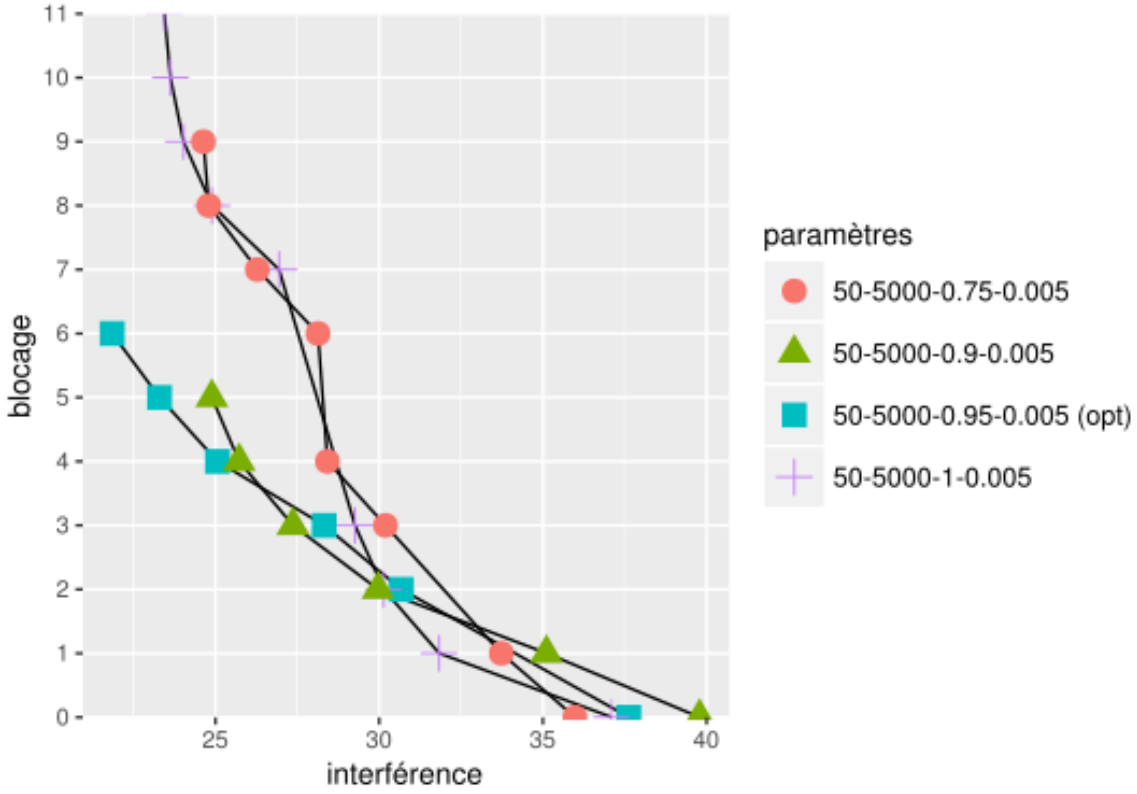


FIGURE 6.5 – Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de croisement de l'algorithme *Nash Pareto GA* sur le benchmark *COST259*

6.3.4 L'algorithme *GCEA*

Nous avons testé les combinaisons de paramètres formés des valeurs présentées dans le tableau 6.5.

TABLEAU 6.5 – Valeurs des paramètres pour l'algorithme *GCEA*

Paramètres	**Valeurs**
Taille de la population	25 ; 50
Nombre de générations	1000 ; 2000 ; 5000
Probabilité de croisement	0.75 ; 0.9 ; 0.95 ; 1
Probabilité de mutation	0.0005 ; 0.005 ; 0.05

Les meilleurs résultats sont donnés dans le tableau 6.6.

La figure 6.6 compare la meilleure combinaison de paramètre trouvée (opt) pour la paire algorithme benchmark *GCEA COST259* par rapport à des combinaisons où seul le nombre de génération et la probabilité de mutation sont modifiés. Remarquons, d'abord, que l'augmentation du nombre de génération n'a pas toujours d'effets sur la qualité des solutions (frontières ● et ■). D'autre part, la diminution de la probabilité de mutation permet d'obtenir de meilleurs résultats (la frontière ■ à 0.005 domine presque toujours la

TABLEAU 6.6 – Meilleurs paramètres pour l'algorithme *GCEA*

Paramètres \ Algorithmes	**GCEA**	
	COST259	Philadelphia
Taille de la population	50	25
Nombre de générations	1000	2000
Probabilité de croisement	0.95	0.95
Probabilité de mutation	0.005	0.005

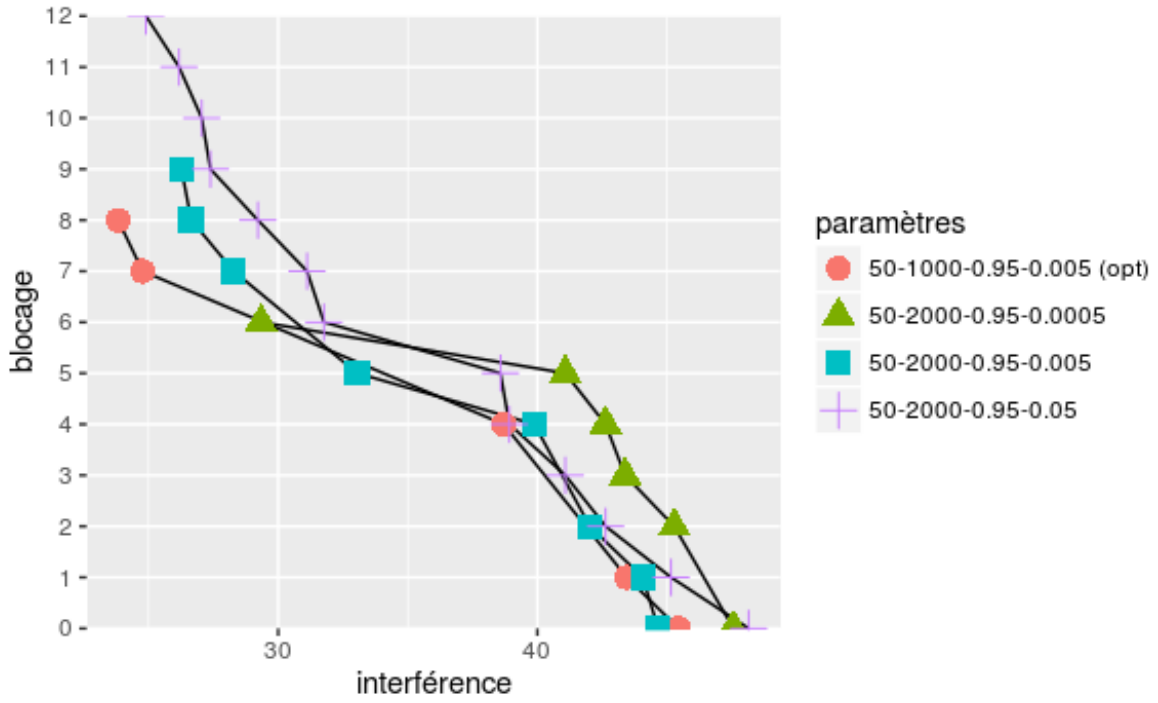


FIGURE 6.6 – Comparaison entre 4 combinaisons de paramètres faisant varier la probabilité de mutation et le nombre de générations de l'algorithme *GCEA* sur le benchmark *COST259*

frontière + à 0.05) mais uniquement jusqu'à un certain seuil à partir duquel le nombre trop peu important de mutations effectuées empêche l'algorithme de progresser (la frontière ■ à 0.005 domine presque toujours la ▲ à à 0.0005).

6.3.5 Discussion des résultats de l'étude empirique

De la série de tests empiriques effectuée, on peut remarquer que :

— L'augmentation de la taille des populations et du nombre de générations améliorent, en général, la qualité des solutions obtenues. Avec, tout de même, une limite, quant au nombre de générations, à partir de laquelle une stagnation semble apparaître ;

— Comme attendu, les taux de mutation faibles et les taux de croisement élevées favorisent l'apparition de solutions de meilleure qualité.

6.4 Étude comparative

Nous allons maintenant comparer les algorithmes entre-eux, chacun avec ses meilleurs paramètres. Nous avons effectué 25 exécutions supplémentaires de chaque algorithme (ce qui fait 30 au total). Puis, nous comparons à l'aide de la métrique C les algorithmes deux à deux en comparant chaque paire d'exécution (chaque exécution de l'un avec chaque exécution de l'autre). Les valeurs obtenues sont illustrées dans la figure 6.7 qui a été générée[7] à l'aide du logiciel libre de traitement et d'analyse de données statistiques `R` et de la bibliothèque `ggplot2`.

Petite indication pour la lecture du tableau : Nous avons, dans chaque cellule, la comparaison entre deux algorithmes. Deux graphiques (boites à moustaches) sont donnés par cellule, un par benchmark. Sur l'axe des ordonnées, la valeur de la métrique C (entre 0 et 1) est donnée. La métrique $C(F1, F2)$ étant principalement une métrique comparant le front (de Pareto) $F2$ au front $F1$, la lecture ce fait colonne par colonne. Plus la valeur sera proche de zéro (plus la boite sera positionnée vers le bas) et plus $F2$ aura l'avantage. Le meilleur algorithme est donc celui qui possède la colonne formée des plus petites valeurs de la métrique.

À la lecture de ces résultats, on peut noter que :

— Les algorithmes *NSGA-II* et *SPEA2* sont quasi-semblables en termes de qualité des solutions, avec un léger avantage pour le premier ;

— L'algorithme *GCEA* semble celui qui donne les résultats les moins bons ;

— L'algorithme *Nash GA* est moyen et a des performances assez variables. Il bat *GCEA* et même *NSGA-II* et *SPEA2* pour le benchmark *Philadelphia* mais se fait écraser par eux pour le benchmark *COST259* ;

— L'algorithme *Nash-Pareto GA* arrive à tirer son épingle du jeu[8]. Il est clairement plus performant sur le benchmark *Philadelphia* et arrive à battre *NSGA-II* et *SPEA2* sur le benchmark *COST259* non sans avoir une certaine variabilité.

Nous comparons aussi notre travail par rapport des travaux récents. Les résultats sont à relativiser du fait que les machines sur lesquelles les algorithmes ont été exécutés diffèrent.

Il s'avère que pour le benchmark *COST259*, qui, rappelons-le, possède énormément de spécificités, aucun des algorithmes que nous avons implémentés n'est parvenu à obtenir des résultats meilleurs que ceux des travaux les plus récents sur toutes les instances. Il serait probablement bénéfique de revoir nos opérateurs génétiques ou d'explorer plus profondément la littérature afin de pouvoir rivaliser. Les bons résultats obtenus sur le benchmark *Philadelphia* ainsi que le fait que même les résultats trouvés par *NSGA-II* et

7. tout comme les graphiques de l'étude empirique
8. C'est le cas de le dire !

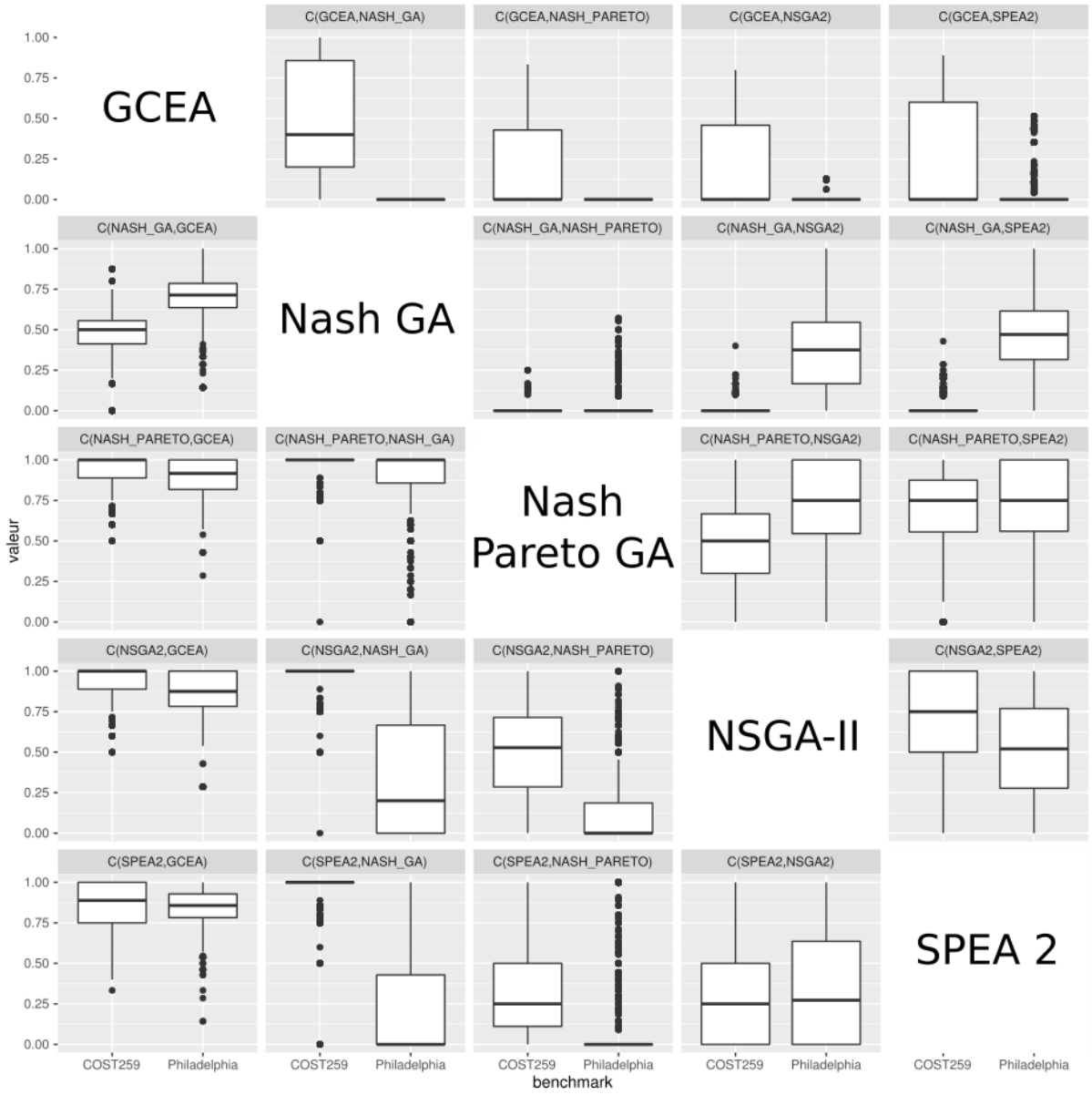


FIGURE 6.7 – Comparaison des algorithmes deux à deux avec 30 exécutions par algorithme

TABLEAU 6.7 – Comparaison de l'algorithme *Nash-Pareto GA* avec des travaux récents sur le benchmark *Philadelphia*

Référence	**Benchmark / Instance**	**Résultat trouvé**	**Notre résultat**
Ghosal2014	Philadelphia / P2	Span=427	Span=**426**
Audhya2013	Philadelphia / P2	Span=449	Span=**426**
Chouiref2012	COST259 / Siemens2	Blocage= 0 ; Interférence=28.67	Blocage= 0 ; Interférence=50.9817
	COST259 / K	Blocage= 0 ; Interférence=36.89	Blocage= 0 ; Interférence=**5.4813**
	COST259 / Swisscom	Blocage= 0 ; Interférence=30.32	Blocage= 0 ; Interférence=34.654
	COST259 / Siemens3	Blocage= 0 ; Interférence=14.25	Blocage= 0 ; Interférence=44.597
Abdessmed2011	COST259 / Siemens1	Blocage= 0 ; Interférence=78.46	Blocage= 0 ; Interférence=**24.7495**
	COST259 / Bradford-nt_1-race	Blocage= 0 ; Interférence=4.43	Blocage= 0 ; Interférence= 25.0379

SPEA2 n'améliorent pas les résultats les plus récents, nous rassurent quant à la validité de nos résultats concernant l'apport de la théorie des jeux et plus particulièrement de l'hybridation Nash-Pareto.

L'équilibre de Nash, via l'algorithme *Nash GA* n'est pas parvenu, seul, à apporté une amélioration des résultats. D'un point de vue théorique, cela est compréhensible du fait que, comme nous l'avons dit au point1.2.2, l'équilibre de Nash ne possède aucune garantie de Pareto-optimalité. En revanche, il peut intervenir, en complémentarité avec la notion de Pareto-optimalité afin de permettre une meilleure exploration de l'espace de recherche et obtenir des solutions satisfaisantes. Les jeux évolutionnaires paraissent être une alternative prometteuse, au vu de nos conclusions il apparaît qu'il soit nécessaire de travailler encore sur l'aspect modélisation afin qu'ils puissent pleinement montrer leur potentiel.

Conclusion

Dans ce chapitre, nous avons décrit la méthodologie que nous avons suivie pour effectuer nos tests. Nous avons, d'abord, déterminer la combinaison de paramètres pour chacun des algorithmes que nous avons implémenté pour utiliser ces paramètres afin de comparer les algorithmes entre eux.

Les algorithmes *Nash GA* et *GCEA* ont des performances moyennes, malgré nos tentatives d'amélioration. L'algorithme *Nash-Pareto GA* est celui qui a obtenu les meilleurs résultats. Il a pu offrir des performances meilleures que les algorithmes *NSGA-II* et *SPEA2*,

montrant ainsi que l'hybridation Nash-Pareto est une voie à suivre pour l'obtention de meilleurs solutions.

Conclusion générale

La théorie des jeux étudie le comportement d'individus confrontés à des situations stratégiques. Ses domaines d'application sont divers et vont de l'économie à l'informatique. Elle propose également des mécanismes permettant d'améliorer les algorithmes d'optimisation multi-objectifs. Nous avons appliqué ses principes afin de résoudre le problème d'affectation de fréquences, problème d'intérêt pratique auquel sont confrontés les opérateurs de téléphonie mobile, et ce, en comparant les approches issues de cette théorie à deux des meilleures méthodes de résolution multi-objectif, à savoir, *NSGA-II* et *SPEA2* qui sont basées sur les algorithmes génétiques. Nous avons effectué la comparaison sur deux benchmarks radicalement différents afin d'assurer une plus grande validité de nos résultats.

Nous avons proposé trois algorithmes se basant sur différents concepts de la théorie des jeux : 1) L'algorithme *Nash GA*, basé sur le travail de Sefrioui et Perlaux (2000) pour la recherche de l'équilibre de Nash. Nous avons améliorer l'algorithme original pour faire en sorte de ne pas perdre les meilleurs solutions durant la résolution ; 2) L'algorithme *Nash-Pareto GA*, basé sur le travail de Lee et al. (2010) une hybridation entre l'équilibre de Nash et le concept de Pareto-optimalité, que nous avons adapté à notre problème ; et 3) l'algorithme *GCEA*, basé sur le travail de Sim et al. (2004) et la recherche de l'équilibre évolutionnaire, que nous avons enrichi, comme pour *Nash GA*, afin de toujours garder les meilleurs solutions trouvées. Nous avons implémenté ces algorithmes en les parallélisant et nous nous sommes appuyés sur la métrique C de Zitzler et al. (2000) afin d'effectuer la comparaison entre les différentes approches.

Il ressort de notre travail que l'équilibre de Nash, seul, n'a pas de réel apport en termes de qualité des solutions et que les jeux évolutionnaires, malgré qu'ils représentent une alternative prometteuse, ne semblent pas encore être tout à fait au point, et ce, malgré que nous ayons tenté d'améliorer le fonctionnement de ces deux approches. En revanche, l'hybridation de l'équilibre de Nash avec des mécanismes de Pareto-dominance permet d'obtenir des améliorations notables. Les méthodes basées sur la théorie des jeux peuvent, donc, venir complémenter ou supplanter les approches classiques de résolution multi-objectif (de type *NSGA-II* et *SPEA2*) dans des contextes pratiques comme celui

du problème d'affectation de fréquences.

Comme perspective, il serait possible de revoir les opérateurs génétiques que nous avons employés afin de mieux les adapter à la résolution du benchmark *COST259*, particulièrement coriace. Aussi, les jeux évolutionnaires gagneraient à être plus étudiés afin de faire émerger leur potentiel. Enfin, il serait envisageable d'employer d'autres mécanismes de la théorie des jeux pour la résolution des problèmes d'optimisation combinatoire multi-objectifs que ce soit en utilisant d'autres équilibres dans les mêmes jeux ou en utilisant d'autres types de jeux.

Bibliographie

K. I. Aardal, S. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, et A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1) :79–129, may 2007.

M. A. Abdessemed. *Résolution du problème d'affectation de fréquences dynamique en utilisant les algorithmes génétiques multi-objectifs parallèles.* Mémoire de fin d'études, École Nationale Supérieure d'informatique (ESI, ex : INI), Alger, jul 2011.

A. Acan, H. Altincay, Y. Tekol, et A. Unveren. A genetic algorithm with multiple crossover operators for optimal frequency assignment problem. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 1, 256–263. IEEE, 2003.

M. Alabau, L. Idoumghar, et R. Schott. New hybrid genetic algorithms for the frequency assignment problem. *IEEE Transactions on Broadcasting*, 48(1) :27–34, mar 2002.

J. M. Alexander. Evolutionary Game Theory. In E. N. Zalta, editor, *Stanford Encyclopedia of Philosophy Evolutionary.* Fall 2009 edition, jan 2002.

J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, et D. Snyers, editors. *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

L. M. Antonio et C. A. C. Coello. A non-cooperative game for faster convergence in cooperative coevolution for multi-objective optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, 109–116. IEEE, may 2015.

G. K. Audhya, K. Sinha, S. C. Ghosh, et B. P. Sinha. A survey on the channel assignment problem in wireless networks. *Wireless Communications and Mobile Computing*, 11(5) : 583–609, may 2011.

T. Başar. Game Theory : Historical Overview. In J. Baillieul et T. Samad, editors, *Encyclopedia of Systems and Control*, 499–504. Springer London, London, 2015.

J. C. Bean. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, 6(2) :154–160, may 1994.

D. Beckmann et U. Killat. A new strategy for the application of genetic algorithms to the channel-assignment problem. *IEEE Transactions on Vehicular Technology*, 48(4) : 1261–1269, jul 1999.

M. Benyagoub et A. Chekal Affari. *Métaheuristiques hybrides à la technique du clustering pour la résolution du problème d'affectation de fréquences*. Mémoire de fin d'études, École Nationale Supérieure d'informatique (ESI, ex : INI), Alger, 2013.

M. Bessedik. *Coloration de graphes par intelligence en essaim*. Thèse de doctorat, École Nationale Supérieure d'informatique (ESI, ex : INI), 2011.

M. Bessedik, A. Daoudi, et K. Benatchba. A Cooperative Approach Using Ants and Bees for the Graph Coloring Problem. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, volume 284, 179–190. Springer International Publishing, 2014.

S. Bleuler, M. Laumanns, L. Thiele, et E. Zitzler. PISA — A Platform and Programming Language Independent Interface for Search Algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, 494–508. Springer Berlin Heidelberg, Faro, Portugal, 2003.

R. Borndörfer, A. Eisenblätter, M. Grötschel, et A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, 76 :73–93, 1998.

P. Bouyaux. *Économétrie Et Théorie Des Jeux*. Base documentaire Applications des mathématiques. Editions Techniques de l'Ingénieur, Rennes, 2014.

D. A. Castañón. Dynamic Noncooperative Games. In *Encyclopedia of Systems and Control*, 323–329. Springer London, London, 2015.

G. Chakraborty et B. Chakraborty. A genetic algorithm approach to solve channel assignment problem in cellular radio networks. In *SMCia/99 Proceedings of the 1999 IEEE Midnight - Sun Workshop on Soft Computing Methods in Industrial Applications (Cat. No.99EX269)*, 34–39. IEEE, 1999.

M. Chakraborty, R. Chowdhury, J. Basu, R. Janarthanan, et A. Konar. A particle swarm optimization-based approach towards the solution of the dynamic channel assignment problem in mobile cellular networks. In *TENCON 2008 - 2008 IEEE Region 10 Conference*, 1–6. IEEE, nov 2008.

D. E. Charilas et A. D. Panagopoulos. A survey on game theory applications in wireless networks. *Computer Networks*, 54(18) :3421–3430, dec 2010.

Y. Chouiref et A. Kaid. *Colonies de fourmis multi-objectifs pour le problème d'affectation de fréquences dynamiques*. Mémoire de fin d'études, École Nationale Supérieure d'informatique (ESI, ex : INI), Alger, 2012.

A. Clarich, E. Rigoni, et C. Poloni. A new Algorithm based on Game Theory for Robust and Fast Multi-Objective Optimisation. Technical report, ESTECO Ltd, Trieste, Italie, jan 2003.

C. Coello. A Short Tutorial on Evolutionary Multiobjective Optimization. *Evolutionary Multi-Criterion Optimization*, 1993(2508) :21–40, 2001.

C. A. C. Coello. A Survey of Constraint Handling Techniques used with Evolutionary Algorithms. *Laboratorio Nacional de Informática Avanzada*, 1999.

Y. Collette et P. Siarry. *Optimisation multiobjectif*. Algorithmes. Eyrolles, Paris, 2002.

C. Cotta et M. Troya. A Comparison of Several Evolutionary Heuristics for the Frequency Assignment Problem. In J. Mira et A. Prieto, editors, *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, volume 2084, 474–481. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

M. Coupechoux et P. Martins. *Vers les systèmes radiomobiles de 4e génération*. Springer Paris, Paris, 2013.

C. Crisan et H. Miihlenbein. The Breeder Genetic Algorithm for Frequency Assignment. In A. E. Eiben, T. Bäck, M. Schoenauer, et H.-P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

M. Cuppini. A genetic algorithm for channel assignment problems. *European Transactions on Telecommunications*, 5(2) :285–294, sep 1994.

C. Daskalakis, P. W. Goldberg, et C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *Communications of the ACM*, 52(2) :89, feb 2009.

K. Deb, A. Pratap, S. Agarwal, et T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, apr 2002.

R. Dorne. An evolutionary approach for frequency assignment in cellular radio networks. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, volume 2, 539–544. IEEE, 1995.

P. B. F. Duarte, Z. M. Fadlullah, A. V. Vasilakos, et N. Kato. On the Partially Overlapped Channel Assignment on Wireless Mesh Network Backbone : A Game Theoretic Approach. *IEEE Journal on Selected Areas in Communications*, 30(1) :119–127, jan 2012.

A. Dupont, A. C. Linhares, C. Artigues, D. Feillet, P. Michelon, et M. Vasquez. The Dynamic Frequency Assignment Problem. *European Journal of Operational Research*, page in press, 2008.

B. Edelman, M. Ostrovsky, et M. Schwarz. Internet Advertising and the Generalized Second Price Auction : Selling Billions of Dollars Worth of Keywords. nov 2005.

A. Eisenblätter, M. C. A. Koster, et M. Grötschel. Frequency Planning and Ramifications of Coloring. *Discussiones Mathematicae Graph Theory*, 22(1) :51–88, 2002.

R. Elsner, M. Maciej, et A. Timm-Giel. *Mobile Networks and Management*, volume 141 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer International Publishing, Cham, 2015.

A. Ghrissi. *Approche intégrative algorithmes génétiques - systèmes immunitaires artificiels pour la résolution du problème d'affectation de fréquences.* Mémoire de fin d'études, École Nationale Supérieure d'informatique (ESI, ex : INI), Alger, 2012.

GSM Association. The Mobile Economy 2015. Technical report, GSM Association, 2015.

D. Haberland. Azyklische Subdigraphenprobleme und Frequenzzuweisung im Mobilfunk. *Master's thesis, Technische Universit at Berlin, Fachbereich Mathematik*, 1996.

H. E. Hadji et M. Babes. Accelerating the convergence of a modified Tabu Search algorithm using a new objective function for the frequency assignment problem. In *2nd International Conference on Systems and Computer Science*, 286–290, aug 2013.

R. Hadji, N. Menni, A. Meraga, et M. Bessedik. Parallel artificial immune system for the constrained graph list multicolouring problem. *International Journal of Metaheuristics*, 3(1) :1, 2014.

W. Hale. Frequency assignment : Theory and applications. *Proceedings of the IEEE*, 68 (12) :1497–1514, dec 1980.

A. Haurie. Cooperative Solutions to Dynamic Games. In *Encyclopedia of Systems and Control*, 235–241. Springer London, London, 2015.

T. D. Hemazro, B. Jaumard, et O. Marcotte. A column generation and branch-and-cut algorithm for the channel assignment problem. *Computers & Operations Research*, 35 (4) :1204–1226, apr 2008.

D. T. Hoang, X. Lu, D. Niyato, P. Wang, D. I. Kim, et Z. Han. Applications of Repeated Games in Wireless Networks : A Survey. *IEEE Communications Surveys & Tutorials*, 17(4) :2102–2135, jan 2015.

J. H. Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

E. Hossain et Z. Han. Game theory for Cognitive Radio Networks Chapter 1 Introduction to Cognitive Radio (CR) Introduction to Cognitive Radio Introduction : Motivation, dec 2009.

S. Hurley, D. H. Smith, et S. U. Thiel. FASoft : A system for discrete channel frequency assignment. *Radio Science*, 32(5) :1921–1939, sep 1997.

M. O. Jackson. A Brief Introduction to the Basics of Game Theory. *SSRN Electronic Journal*, 2011.

B. Jaumard, O. Marcotte, C. Meyer, et T. Vovor. Erratum to "Comparison of column generation models for channel assignment in cellular networks". *Discrete Applied Mathematics*, 118(3) :299–322, may 2002.

M.-H. Jin, H.-K. Wu, J.-T. Horng, et C.-H. Tsai. An evolutionary approach to fixed channel assignment problems with limited bandwidth constraint. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, volume 7, 2100–2104. IEEE, 2001.

R. Józefowicz, I. Poźniak-Koszalka, L. Koszalka, et A. Kasprzak. *Recent Developments in Computational Collective Intelligence*, chapter Algorithms, 135–144. Springer International Publishing, Cham, 2014.

A. Kaci et O. Ait El Hara. *Approche par classification des variables pour la résolution de problèmes d'optimisation combinatoire : Application au problème Weighted MAX-SAT et au Problème d'Affectation de Fréquences*. Mémoire de fin d'études, École Nationale Supérieure d'informatique (ESI, ex : INI), Alger, 2011.

H. Kamal, M. Coupechoux, et P. Godlewski. Inter-operator spectrum sharing for cellular networks using game theory. In *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, 425–429. IEEE, sep 2009.

H. Kamal, M. Coupechoux, et P. Godlewski. Tabu search for dynamic spectrum allocation in cellular networks. *Transactions on Emerging Telecommunications Technologies*, 23 (6) :508–521, oct 2012.

D. Kumar Singh, K. Srinivas, et D. Bhagwan Das. A Useful Metaheuristic for Dynamic Channel Assignment in Mobile Cellular Systems. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(6) :6, 2012.

W. Lai et G. Coghill. Channel assignment through evolutionary optimization. *IEEE Transactions on Vehicular Technology*, 45(1) :91–96, 1996.

X. Lai, Q. Liu, W. Wang, L. Li, S. Lu, et Y. Zhao. Dynamic game with perfect and complete information based dynamic channel assignment. *Applied Intelligence*, 1–13, 2012.

D. Lee, J. Periaux, et L. F. Gonzalez. UAS Mission Path Planning System (MPPS) Using Hybrid-Game Coupled to Multi-Objective Optimizer. *Journal of Dynamic Systems, Measurement, and Control*, 132(4) :041005, 2010.

C. E. Lemke et J. T. Howson. Equilibrium Points of Bimatrix Games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2) :413–423, 1964.

K. Leyton-Brown et Y. Shoham. Essentials of Game Theory : A Concise Multidisciplinary Introduction. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2(1) : 1–88, jan 2008.

Z. Li, S. Liu, D. Xiao, J. Chen, et K. Li. Multi-objective particle swarm optimization algorithm based on game strategies. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation - GEC '09*, page 287, New York, New York, USA, 2009. ACM Press.

A. Mahmoudi. *Une approche basée sur les systèmes immunitaires artificiels pour la résolution du problème de la T-Coloration des graphes : Application aux problèmes d'affectation de fréquences.* Mémoire de magister, École Nationale Supérieure d'informatique (ESI, ex : INI), 2011.

S. Matsui et K.-i. Tokoro. A new genetic algorithm for minimum span frequency assignment using permutation and clique. In *Genetic and Evolutionary Com- putation Conference 2000 (GECCO-2000)*, 682–689, 2000.

S. Matsui, I. Watanabe, et K.-i. Tokoro. A Parameter-Free Genetic Algorithm for a Fixed Channel Assignment Problem with Limited Bandwidth. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, 789–799. Springer-Verlag, sep 2002.

S. Matsui, I. Watanabe, et K.-i. Tokoro. An efficient hybrid genetic algorithm for a fixed channel assignment problem with limited bandwidth. *Lecture notes in computer science*, 2724, 2003.

S. Matsui, I. Watanabe, et K.-I. Tokoro. Application of the parameter-free genetic algorithm to the fixed channel assignment problem. *Systems and Computers in Japan*, 36 (4) :71–81, apr 2005.

N. Megiddo et C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2) :317–324, apr 1991.

R. Meng, Y. Ye, et N.-g. Xie. Multi-objective optimization design methods based on game theory. In *2010 8th World Congress on Intelligent Control and Automation*, 2220–2227. IEEE, jul 2010.

B. Metzger. Spectrum management technique. In *38th national ORSA meeting*, 1970.

H. Meunier. *Algorithmes Évolutionnaires Parallèles Pour L'optimisation Multi-Objectif De Réseaux De Télécommunications Mobiles.* PhD thesis, Université Des Sciences Et Techologies De Lille, 2002.

H. E. Mezhoudi. *Résolution du problème d'affectation de fréquence dynamique en utilisant un algorithme basé sur le comportement des abeilles*. Mémoire de fin d'études, École Nationale Supérieure d'informatique (ESI, ex : INI), Alger, 2010.

J. F. Nash. Equilibrium Points in N-Person Games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1) :48–49, jan 1950.

C. Ngo et V. Li. Fixed channel assignment in cellular radio networks using a modified genetic algorithm. *IEEE Transactions on Vehicular Technology*, 47(1) :163–172, 1998.

M. J. Osborne et A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, Massachusetts, feb 1994.

A. Ozdaglar. Strategic Form Games and Nash Equilibrium. In *Encyclopedia of Systems and Control*, 1364–1372. Springer London, London, 2015.

C. H. Papadimitriou. On the Complexity of the Parity Argument and other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3) :498–532, 1994.

J. Periaux, F. Gonzalez, et D. S. C. Lee. *Evolutionary Optimization and Game Strategies for Advanced Multi-Disciplinary Design*, volume 75 of *Intelligent Systems, Control and Automation : Science and Engineering*. Springer Netherlands, Dordrecht, 2015.

S. Pinagapany et A. V. Kulkarni. Solving channel allocation problem in cellular radio networks using genetic algorithm. *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, 239–244, 2008.

B. Polak. Open Yale Courses, Econ 159 : Game Theory, Lecture 11 - Evolutionary Stability : Cooperation, Mutation, and Equilibrium, apr 2007.

M. Potter et K. D. Jong. A cooperative coevolutionary approach to function optimization. *Parallel Problem Solving from Nature*, 249 – 257, 1994.

T. Roughgarden. *Selfish routing*. PhD thesis, Cornell University, jan 2002.

T. Roughgarden. Computing equilibria : A computational complexity perspective. *Economic Theory*, 42(1) :193–236, feb 2009.

T. Roughgarden. Lectures Notes on Algorithmic Game Theory (CS364A), Fall 2013, 2013.

M. Sefrioui et J. Perlaux. Nash genetic algorithms : examples and applications. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, 509–516. IEEE, 2000.

L. S. Shapley. A Value for n-person Games. Technical report, DTIC Document, 1952.

A. Shukla, R. Tiwari, S. Rungta, et M. S. Kumar. A New Heuristic Channel Assignment in Cellular Networks. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 7, 473–478. IEEE, 2009.

K.-B. Sim, D.-W. Lee, et J.-Y. Kim. Game Theory Based Coevolutionary Algorithm : A New Computational Coevolutionary Approach. *International Journal of Control, Automation, and Systems*, 2(4) :463–474, 2004.

J. Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

J. M. Smith. Evolution and the Theory of Games : In Situations Characterized by Conflict of Interest, the Best Strategy to Adopt Depends on What Others Are Doing. *American Scientist*, 61(1) :41–45, 1976.

K. Smith. A genetic algorithm for the channel assignment problem. In *IEEE GLOBECOM 1998 (Cat. NO. 98CH36250)*, volume 4, 2013–2018. IEEE, 1998.

R. Srikant. Nash equilibrium. In *Encyclopedia of Systems and Control*, 831–835. Springer London, London, 2015.

M. Srinivas et L. Patnaik. Genetic algorithms : a survey. *Computer*, 27(6) :17–26, jun 1994.

S. Suliman, G. Kendall, et I. Musirin. Optimizing channel allocation in wireless communication using single-swap mutation based heuristic. *International Conference on Advanced Communication Technology, ICACT*, 774–778, 2013.

S. I. Suliman, G. Kendall, et I. Musirin. Artificial immune algorithm in solving the channel assignment task. In *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, number November, 153–158. IEEE, nov 2014.

J. Szymanik. Backward Induction Is PTIME-complete. In D. Grossi, O. Roy, et H. Huang, editors, *Proceedings of the Fourth International Workshop on Logic, Rationality and Interaction, Lecture Notes in Computer Science*, volume 8196 of *Lecture Notes in Computer Science*, 352–356. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

P. D. Taylor et L. B. Jonker. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 40(1-2) :145–156, jul 1978.

C. Valenzuela, S. Hurley, et D. Smith. A permutation based Genetic Algorithm for minimum span frequency assignment. In A. E. Eiben, T. Bäck, M. Schoenauer, et H.-P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, 907–916. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

C. L. Valenzuela. A Study of Permutation Operators for Minimum Span Frequency Assignment Using an Order Based Representation. *Journal of Heuristics*, 7(1) :5–21, 2001.

J. Von Neumann et O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton university press, 1944.

L. Wang, S. N. S. Lee, et W. Y. Hing. Solving channel assignment problems using local search methods and simulated annealing. In H. Szu, editor, *Proceedings of SPIE, Independent Component Analyses, Wavelets, Neural Networks, Biosystems, and Nanoengineering IX*, may 2011.

X. Wang, Q. Han, X. Guan, et K. Ma. Price-based interference management in dense femtocell systems. *International Journal of Communication Systems*, 28(1) :19–37, jan 2015.

J. W. Weibull. *Evolutionary Game Theory*. MIT Press, 1997.

M. Xiao, X. Shao, L. Gao, et Z. Luo. A new methodology for multi-objective multidisciplinary design optimization problems based on game theory. *Expert Systems with Applications*, 42(3) :1602–1612, 2015.

Y. Xu et I. Sakho. *Frequencies Assignment in Cellular Networks*, chapter Frequencie, 211–220. Springer International Publishing, Cham, 2015.

F. Yu, A. Bar-Noy, P. Basu, et R. Ramanathan. Algorithms for channel assignment in mobile wireless networks using temporal coloring. In *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems - MSWiM '13*, 49–58. ACM Press, 2013.

Y. Zhang et M. Chen. A Metaheuristic Approach for the Frequency Assignment Problem. In *2010 International Conference on Computational Intelligence and Software Engineering*, 1–5. IEEE, sep 2010.

E. Zitzler, K. Deb, et L. Thiele. Comparison of Multiobjective Evolutionary Algorithms : Empirical Results. *Evolutionary Computation*, 8(2) :173–195, jun 2000.

E. Zitzler, M. Laumanns, et L. Thiele. SPEA2 : Improving the Strength Pareto Evolutionary Algorithm. Technical report, Swiss Federal Institute of Technology (ETH), Zurich, 2001.