Numéro d'ordre :69/2019-C/INF

République Algérienne Démocratique et Populaire
Université Des Sciences et de la Technologie Houari Boumedienne
Faculté d'Electronique et d'Informatique

Thèse de DOCTORAT
présentée pour obtenir le grade de docteur

En : Informatique
Spécialité : Intelligence Artificielle.

PAR
Yasmine LAHSINAT

Thème

Résolution métaheuristique du problème d'affectation de fréquences dans les réseaux cellulaires.

Soutenue publiquement le 23/07/2019 devant le jury composé de :

M.BOUKRA Abdelmadjid	Professeur	à l'USTHB	Président
Mme.BOUGHACI Dalila	Professeur	à l'USTHB	Directrice de thèse
M.BENHAMOU Belaid	Professeur	à l'AMU	Co-Directeur de thèse
M.LEBAH Yahia	Professeur	à l' Univ Oran	Examinateur
Mme.SI TAYEB Fatima	Professeur	à l'ESI	Examinateur
M.DAOUDI Mourad	Maitre de conférences	à l'USTHB	Examinateur

Remerciements

Je tiens tout d'abord à remercier chaleureusement ma directrice de thèse : le Pr Dalila BOUGHACI pour tout le soutien, les directives, encouragements ainsi que les précieux conseils qu'elle m'a prodigués tout au long de ces années de thèse.
Je remercie également mon Co-directeur de thèse pour m'avoir reçu lors de mon stage à Marseille.
Je tiens particulièrement à remercier le président du jury le Pr. BOUKRA pour l'honneur qu'il me fait de présider ma thèse.
Je remercie le président du jury ainsi que les différents membres pour l'honneur qu'ils me font en consacrant de leur temps afin d'évaluer ce travail. Je les remercie également pour leur précieuses recommandations et remarques constructives.

Je termine enfin en remerciant ceux sans qui je n'existerais pas, ceux pour qui je vis, ceux sans qui je ne serais rien aujourd'hui. Alors un grand merci à mes parents et à ma sœur Nesrine, qui m'ont épaulé, soutenu, financé, encouragé au cours de cette laborieuse aventure qui est celle de la préparation d'une thèse de doctorat.

Résumé

Dans cette thèse nous nous intéressons au problème de minimisation d'Interférences lors de l'Affectation de Fréquences (en anglais the mimnimum interference frequency assignment problemé (MI-FAP)) dans un réseau cellulaire. Le problème d'affectation de fréquences consiste à attribuer des fréquences à chaque émetteur/récepteur du réseau de sorte que deux émetteurs/récepteurs proches géographiquement n'aient pas la même fréquence, ou des fréquences proches. Ces fréquences doivent respecter une contrainte de séparation afin d'obtenir une affectation de fréquences qui ne génère pas ou génère le moins d'interférences possible. Le problème que nous traitons est un problème d'optimisation combinatoire bien connu. Le MI-FAP est une généralisation du problème de coloration de graphe et de ce fait considéré comme un problème NP-difficile. Nous proposons de résoudre ce problème en utilisant les métaheuristiques. En effet ces dernières sont un outil très puissant pour résoudre les problèmes d'optimisation combinatoire et offrent un bon rapport qualité/temps.
Nous retenons trois métaheuristiques différentes : Harmony search, Jaya et la recherche à voisinage variable. Nous adaptons dans un premier temps ces métaheurisitques. Puis nous apportons à ces méthodes certains changements pour pallier les inconvénients de la convergence lente et le manque de diversité remarquée.
Nous proposons également une résolution basée les hypeheuristique. Nous adoptons alors trois stratégies différentes : l'hyperheurisitque aléatoire, l'hyperheuristiques à fonction de choix et la stochastique hyperheuristique.
Les résultats obtenus sur les différentes instances du problème montrent l'efficacité des stratégies mises en œuvre par les différents algorithmes que nous avons proposés.

Mots clés : MI-FAP, métaheurisitque, optimisation, interférence, HS,Jaya, VNS.

Abstract

In this PhD thesis we review the Minimum Interference Frequency Assignment Problem (MI-FAP) in a radio network. The MI-FAP which is a major issue in the radio networks as well as a well-known NP-hard combinatorial optimization problem. The challenge is to assign a frequency to each transceiver (TRX) of the network with limited or no interferences at all. Indeed, considering that the number of radio networks users is ever increasing and that the radio spectrum is a scarce and expensive resource, the latter should be carefully managed in order to avoid any interference. In this respect, we propose to solve the MI-FAP using metaheuristics just like many other researchers. Indeed the metaheuristics have emerged as powerful means to solve hard combinatorial optimization problem. We favoured three metaheuristics: Harmony search, jaya and the variable neighborhood search. We were eager to explore in depth their specifications and to investigate their ability to solve the MI-FAP. Moreover, it was an appealing challenge for us to propose an effective enhancement for the three algorithms designed to deal with the MI-FAP. We therefore proposed two enhancements of the VNS meant to improve its performances. The proposed enhancements involved the use of some features pertaining to other successful single based metaheuristics, namely the iterated local search and the Breakout local search. Also, we propose two enhancements of the HS and two other for the Jaya algorithm. The perturbation schemes and operators proposed to enhance the methods will also be thoroughly reviewed. We also propose three hypeheuristics to deal with the MI-FAP namely the choice function hyperheuristic, the random hyperheuritsic and the stochastic hyperheuristcis. The different results are encouraging.
Our experiments showed clearly that the proposed enhancements can achieve better results than the basic algorithms. The different enhanced methods appear to be a promising approaches.

Keywords: MI-FAP, metaheuristics, optimisation, interférencen, HS, jaya, VNS.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La gestion efficiente de la ressource radio est une difficulté réelle et un enjeu majeur pour les opérateurs de téléphonie mobile. Elle constitue une problématique à part entière (connu sous l'appellation du problème d'affectation de fréquences) qui survient dans le domaine des télécommunications et qui est dû notamment à l'essor fulgurant de la téléphonie mobile au cours de ces 20 dernières années. L'affectation de fréquences est une phase clé dans le design d'un réseau téléphonique. En effet, la gestion du spectre radio influence la qualité de service de l'opérateur. Cette ressource étant rare et coûteuse, son utilisation de manière optimale est nécessaire. En effet, le nombre sans cesse grandissant des utilisateurs, implique plus d'émetteurs/récepteurs communicant via une ressource radio. Or le nombre de fréquences est limité, ce qui nécessite le recours au principe de réutilisation de fréquences. Ce procédé n'a pas que des avantages puisque l'observation de son utilisation révèle la naissance d'interférences qui peuvent avoir un impact négatif sur le réseau. Ces interférences sont engendrées dans le cas où des émetteurs/récepteurs proches géographiquement utilisent des fréquences similaires, adjacentes ou pas suffisamment distantes. Il est alors question de manager ces fréquences et ces émetteurs/récepteurs de manière à éviter les interférences. C'est ce qui est désigné sous l'appellation " problème de minimisation d'interférences lors de l'affectation de fréquences".

La minimisation d'interférences lors de l'affectation de fréquences est un problème d'optimisation combinatoire bien connu de la communauté des chercheurs. Il est de la classe des problèmes de coloration de graphe et à ce titre, il est démontré NP-Difficile. Aussi, cette problématique fait appel aux possibilités offertes par l'intelligence artificielle. C'est pourquoi notre choix s'est porté sur les métaheuristiques qui sont un outil des plus intéressants pour résoudre ce problème. Si notre démarche s'inscrit dans ce cadre, il reste que notre ambition est de proposer des algorithmes efficaces pour résoudre le MI-FAP.

La première partie de cette thèse comporte deux chapitre. Un premier chapitre contenant une brève introduction à l'optimisation combinatoire et présenterons les principales métaheuristiques. Le second chapitre traitera le problème d'affectation de fréquences, ses spécificités et comprendra une revue de la littérature concernant les différentes méthodes proposées pour résoudre des instances du MI-FAP et sa formulation mathématique.

La seconde partie de cette thèse consistera en une présentation de toutes les contributions que nous avons faites au cours de ces années de recherche notamment : le travail dédié à la recherche à voisinage variable, la recherche harmonique, l'algorithme de jaya et celui consacré aux hyperheuristiques. Nous présenterons l'ensemble de ces méthodes ainsi que l'adaptation faite de ces méthodes aux instances du problème MI-FAP traitées. Nous présenterons également les changements apportés à ces différentes méthodes dans le but d'améliorer les résultats, de même que les résultats obtenus suite aux expérimentations réalisées. Nous ferons une synthèse des performances réalisées pour chacune des méthodes mentionnées.

Le manuscrit est scindé en deux parties :

La première partie consacrée à l'état de l'art se décline en deux chapitres :
Le premier chapitre : Introduction aux concepts de base liés à l'optimisation combinatoire et aux métaheuristiques.
Le second chapitre : Introduction, Bibliographie et Présentation du problème MI-FAP.
La seconde partie du manuscrit consacrée aux contributions s'articule comme suit :
Le premier chapitre : Présentation des travaux réalisés sur la métaheuristique à voisinage variable et Analyse des différents résultats obtenus sur les différentes instances du MI-FAP.
Le deuxième chapitre : Présentation des travaux réalisés sur la métaheuristique harmony search et Analyse des résultats enregistrés sur les différentes instances du MI-FAP.
Le troisième chapitre : Présentation des travaux dédiés à la métaheurisitque Jaya et Analyse des résultats obtenus sur les différentes instances du MI-FAP.
Le quatrième chapitre : Présentation des travaux dédié aux hyperheuristiques et Analyse des résultats obtenus sur les différentes instances du MI-FAP.

Première partie

Etat de l'art

1. Optimisation combinatoire et méta-heuristiques

Sommaire

1.1. Introduction

Ce chapitre a pour objet d'introduire quelques notions et définitions élémentaires relatives à la théorie de l'optimisation et la recherche opérationnelles. Ces dernières sont nécessaires à la compréhension du problème MI-FAP qui est défini formellement dans le cadre d'un problème d'optimisation. Plus particulièrement comme un problème d'optimisation combinatoire. Nous commencerons donc ce chapitre par donner la définition d'un problème d'optimisation combinatoire. Nous citerons ensuite quelques problèmes d'optimisation combinatoire bien connus. Nous aborderons brièvement la complexité de résolution de ces problèmes. Nous verrons en détail les deux principales classes de méthodes de résolution des problèmes d'optimisation combinatoires. Enfin, nous donnerons un aperçu des principales métaheuristiques.

1.2. Définition des problèmes d'optimisations

Très souvent, les ingénieurs se heurtent à des problèmes de complexité grandissante qui surgissent dans différents domaines de recherche, et qui peuvent être fréquemment exprimés sous forme d'un problème d'optimisation, que l'on cherche à optimiser (maximiser ou minimiser). En nous appuyant sur les ouvrages de Sakarovitch[67], Papadimitriou [62],[9] et [13], nous proposons dans cette section une introduction non exhaustive des définitions de base de la théorie de l'optimisation.

1.2.1. Problèmes d'optimisation

Un problème d'optimisation est généralement caractérisé par un ensemble de variables, une fonction objectif f, un espace de recherche Set un ensemble de contraintes d'égalités (ou inégalités) que les variables doivent satisfaire. La fonction objectif f associe une valeur à chaque solution. Selon le problème que l'on désire résoudre, il est alors question de trouver la (ou les) solution(s) s^* de S minimisant ou maximisant la fonction f.
De manière plus formelle [9] : un problème d'optimisation, noté $P(X, f)$, est caractérisé par un ensemble réalisable ou admissible X non-vide et une fonction objectif f qui associe un scalaire dans R à chaque élément x de l'ensemble X. Les éléments de X sont dits solutions réalisables. Résoudre le problème $P(X, f)$ revient à trouver parmi les solutions réalisables, une qui minimise ou maximise f, c'est-à-dire dans le cas d'un problème de minimisation, trouver une solution $x^* \in X$ telle que $f(x) \geq f(x^*)$ pour tout élément x dans X. Une telle solution est dite optimale et sera notée $x(X, f)$.

1.2.2. Problèmes d'optimisation discrète

Il existe différents problèmes d'optimisation susceptibles d'être classés selon la fonction objectif ou selon les variables. Si on considère la fonction objective on distingue alors entre les problèmes multi-objectif (i.e plusieurs fonctions à optimiser) et les problèmes monoobjectif (une seule fonction à optimiser). Il est également possible de distinguer entre l'optimisation dynamique ou statique (i.e. la fonction objective change avec le temps). Si on considère les variables, on peut alors distinguer entre les problèmes d'optimisation à variables discrètes ou les problèmes d'optimisation à variables continues.
Les problèmes d'optimisations discrètes sont une classe de problèmes particulièrement étudiée. Nous donnons la définition proposée dans [9] pour les variables de ce type de problèmes : Définition : Tout ou partie des variables de ce type de problèmes appartiennent à l'ensemble des entiers, autrement dit $X \subseteq Z^m \times R^{n-m}$, avec

$$0 \leq m \leq n$$

. Lorsque $m = n$, $P(X, f)$ est dit problème en nombres entiers, sinon il fait partie des problèmes d'optimisations mixtes en nombres entiers. Bien que l'ensemble réalisable soit plus restreint dans le cas de l'optimisation discrète, ces problèmes sont souvent plus difficiles que leurs versions continues.

1.2.3. Problèmes d'optimisation combinatoire

Les problèmes d'optimisation combinatoire sont des problèmes d'optimisation dont les ensembles réalisables sont finis mais combinatoires[9].
Selon Sakarovitch[67] un problème d'optimisation combinatoire peut être formulé de la façon suivante : Soit S l'ensemble des solutions possibles d'un problème. Soit $X \subset S$ l'ensemble des solutions admissibles vérifiant un ensemble de contraintes C. Soit $f : X -> R$, une fonction que l'on nomme fonction objectif. Un problème d'optimisation se définit par : $minf(x)$, $x \in X$. Lorsque l'ensemble S des solutions est discret on parle de problème d'optimisation combinatoire.
Le problème consistant à chercher un élément maximum au lieu d'un élément minimum est de même nature puisque : $Max[f(s)] = -Min[-f(s)]$.
Aussi, un problème d'optimisation se définit par l'ensemble de ses instances, souvent infiniment nombreuses. Dans la pratique, le problème se ramène à résoudre numériquement l'une de ces instances, par un procédé algorithmique. A chaque instance du problème est associé un ensemble de solutions potentielles, qu'on appellera X, qui est fini et de grande taille. L'ensemble des solutions potentielles est généralement soumis à des contraintes. On associe à un problème d'optimisation combinatoire une fonction f : dite fonction objectif ou fonction coût qui servira à évaluer la qualité de chaque solution. Résoudre un problème d'optimisation combinatoire consiste alors à trouver une solution $s \in X$ optimisant la valeur de la fonction coût f et vérifiant les contraintes : on dira donc qu'elle est réalisable. Formellement un problème d'optimisation combinatoire se caractérise comme suit :

- un ensemble de variables $(x_1, x_2, \ldots, x_n)$.
- un ensemble de contraintes C sur les variables : par exemple, des équations ou des inéquations etc...
- S ensemble de solutions possibles : $S = s = (x_1, x_2, \ldots, x_n)$
- Une application $f : X -> R$.

Il s'agit alors de déterminer $s^* \in S$ tel que $f(s^*) < f(s)$ pour tout $s \in X$ tels que s^* satisfasse toutes les contraintes [62]. Une telle solution s^* s'appelle une solution optimale, ou un optimum global.

1.3. La complexité

L'optimisation combinatoire consiste à trouver un optimum parmi un nombre fini de choix en minimisant (ou maximisant) une fonction, avec ou sans contraintes.

Quand le nombre de combinaisons possibles devient exponentiel par rapport à la taille du problème, le temps de calcul devient rapidement problématique que pour certains problèmes on ne connaît pas d'algorithme exact polynomial, permettant de trouver cet optimum, en un temps de calcul polynomial, soit un temps proportionnel à N^n , où N désigne le nombre de paramètres inconnus du problème et où n est une constante entière [19]. Lorsqu'on conjecture qu'il n'existe pas une telle constante n telle qu'un polynôme de degré n puisse borner le temps de calcul d'un algorithme, on parle alors d'optimisation difficile, ou de problèmes NP-difficiles (c'est tout l'objet de la théorie de la NP-complétude) [5]. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données [33].
A l'heure actuelle, la théorie de la complexité des algorithmes montre que la plupart des problèmes d'optimisation combinatoire sont NP-complet, c'est-à-dire qu'on ne connaît pas d'algorithme polynomial pour les résoudre. La difficulté intrinsèque de ces problèmes est bien caractérisée par la théorie de la NP-complétude qui n'est pas seulement théorique mais se confirme hélas dans la pratique [33]. Il arrive que des algorithmes exacts de complexité exponentielle se comportent efficacement face à de très grosses instances - pour certains problèmes et certaines classes d'instances[41]. Mais c'est très souvent l'inverse qui se produit ; pour de nombreux problèmes, les meilleures méthodes exactes peuvent être mises en échec par des instances de taille modeste, parfois à partir de quelques dizaines de variables seulement. Ceci est un modeste résumé sur la théorie de la complexité pour davantage d'information se référer à [62].

1.4. Méthodes de résolutions

Depuis des années, les chercheurs se sont attelés, à la dure tâche de concevoir des méthodes performantes pour résoudre les problèmes d'optimisation combinatoire, dont l'objectif est de trouver la meilleure solution réalisable, dans un ensemble de solutions fini mais souvent de cardinalité très grande. Dés lors, deux grandes familles de méthodes se distinguent : les méthodes exactes et les méthodes approchées. On trouve dans la première famille la plupart des méthodes traditionnelles développées depuis une trentaine d'années telles que : l'algorithme du simplexe et le dual. Ces dernières ont permis de trouver des solutions optimales à des problèmes de tailles raisonnables. Malgré les progrès réalisés, les méthodes exactes ne sont pas toujours efficaces : elles rencontrent généralement des difficultés avec les applications de taille importante ou le temps de calcul devient très vite problématique, que pour certains problèmes on parle d'optimisation difficile ou de problèmes dit NP-difficile. Résoudre les problèmes

dits NP complet efficacement constitue un intérêt croissant pour les chercheurs. C'est pourquoi le domaine de l'optimisation combinatoire a connu et connaît toujours le développement de méthodes de la deuxième famille, à savoir les méthodes approchées dites Méta-heuristiques. Ces dernières permettent de trouver un optimum parmi un nombre fini de choix, souvent très grand, en un temps de calcul raisonnable, elles font alors face à l'explosion combinatoire dont souffrent les méthodes dites exactes.
Dans ce qui suit nous donnons un aperçu des principales méthodes relatives aux méthodes exactes et approchées.

1.4.1. Les méthodes exactes

Elles permettent de trouver un optimum globale et ce en faisant une recherche exhaustive par énumération explicite des solutions potentielles, de l'espace de recherche, par exemple : la méthode de séparation et évaluation « Branch and Bound » (voir [44]) Cette dernière s'effectue comme suit : L'espace de recherche est divisé en sous-espaces, puis on cherche une borne minimale en termes de fonction objectif, associée à chaque sous-espace de recherche. Les "mauvais" sous-espaces seront éliminés. Les étapes précédentes seront reproduites jusqu'à l'obtention de l'optimum global. Cette méthode présente comme inconvénient l'impossibilité d'être appliquée à des problèmes dont l'espace de recherche est de taille trop importante et ce en raison du temps de calcul induit qui augmente exponentiellement avec la taille du problème. Prenant comme exemple le problème du voyageur de commerce sur un réseau comportant 100 villes. Dire qu'il suffit de faire la liste des tournées et d'en choisir celles dont la longueur est minimum, serait une erreur majeure vu le nombre astronomique de possibilités. L'espace de recherche croît en $(n-1)!$, où « n » est le nombre de villes à visiter, ce qui dépasse rapidement les capacités de calcul de n'importe quel ordinateur. Avec seulement 50 villes, il faudrait évaluer 49 ! trajets.C'est l'explosion combinatoire ! D'autres méthodes exactes telles que A* et bien d'autres, ont certes permis de trouver des solutions dont l'optimalité est garantie, mais pour des problèmes de taille raisonnable et malgré les progrès réalisés, notamment en matière de programmation linéaire en nombres entiers.Ces méthodes rencontrent toujours des difficultés face au temps de calcul et à la dimension du problème, elles restent donc impuissantes devant l'explosion combinatoire. De plus, la majorité des situations rencontrées dans la pratique, présentent des complications, qui mettent en défaut ces méthodes. Notamment lorsque la fonction objectif n'est pas linéaire, ou dans le cas où elle ne peut pas s'exprimer analytiquement en fonction des paramètres, ou encore, dans le cas où le problème à résoudre exige la considération simultanée de plusieurs objectifs contradictoires. D'où le besoin d'autres méthodes qui forment la deuxième catégorie à savoir les méthodes approchées[5].

1.4.2. Les méthodes approchées

Ces méthodes constituent une alternative très intéressante par rapport à la première catégorie, en raison de leur application aux problèmes d'optimisation difficiles : la taille du problème n'est plus une contrainte. Ces méthodes assurent une solution de bonne qualité sans garantie de l'optimalité mais au profit d'un temps de calcul raisonnable. Nous détaillons dans ce qui suit l'essentiel de ces méthodes. On peut citer les méthodes gloutonnes et l'amélioration itérative : par exemple, la méthode de Lin et Kernighan qui resta longtemps le numéro un des algorithmes pour le problème du voyageur de commerce [50].

1.4.2.1. Les heuristiques

Heuristique : du grec heuriskein qui signifie trouver (heurica). Dans le contexte actuel, ce terme est synonyme de toutes les méthodes approximatives développées dans le but d'accélérer la recherche d'une solution à un problème. Dans la littérature une heuristique est définie comme étant une astuce, une stratégie qui permettrait de limiter rigoureusement la recherche, dans l'espace de solutions possibles ainsi un gain de temps est assuré mais en contrepartie, les heuristiques ne garantissent pas la qualité de la solution. Ceci n'est pas vraiment un problème car il arrive souvent que pour certains problèmes, une bonne solution trouvée en un temps raisonnable soit préférable. Elles sont « heuristiques » au sens où elles ne garantissent pas d'atteindre une solution optimale mais ont pour ambition de fournir une « bonne » solution approchée en un temps de calcul « raisonnable ».

1.4.2.2. Les métaheuristiques

Les métaheuristiques sont des méthodes approchées, non déterministes incorporant le plus souvent un principe stochastique, conçues dans le but de résoudre des problèmes ayant des instances de grandes tailles, demandant un calcul scientifique intensif, généralement pour résoudre des problèmes d'optimisation difficile et ce afin de faire face à l'explosion combinatoire. Elles se placent à un niveau plus général ; en effet il n'est pas toujours évident de trouver l'heuristique appropriée ; cette dernière dépendant en grande partie du domaine d'application et très spécifique au problème traité et nécessite pour son développement des connaissances avancées. Une métaheuristique est donc une méthode flexible, très générale [13], c.à.d. avant de pouvoir être appliquée à la résolution d'un problème particulier, quelques transformations (mineures en général) sont nécessaires. L'ajout du préfixe « méta » signifie que le champ d'application des métaheuristiques ne se limite pas à priori à un problème donné mais doit pouvoir s'étendre à de nombreux problèmes variés pour lesquels elles s'adapteront avec plus ou moins de facilité et de bonheur [70]. Une métaheuristique doit donc être adaptable afin de convenir aux exigences d'un problème bien spécifique.

1.4.2.3. Classification des méta-heuristiques (toutes les classifications possibles)

Plusieurs métaheuristiques ont été développées à ce jour assurant ainsi la résolution de problèmes d'optimisations démontrés difficiles. Différentes manières de classer ces dernières sont à distinguer. La première consiste à distinguer entre les métaheuristiques s'inspirant de phénomènes naturels. En effet nombreuses métaheuristiques s'inspirent de la nature tel que recuit simulé qui s'inspire de la physique, d'autres s'inspirent de la biologie (algorithmes génétiques, recherche tabou, etc...) ou de l'éthologie (colonies de fourmis, essaims de particules, etc...) ou même de la biogéographie. En face nous avons des méthodes qui ne font pas d'analogie à la nature, on citera : la méthode de descente, la recherche à voisinage variable.
Une autre approche permet de distinguer entre les métaheuristiques travaillant avec une population de solutions (elles sont connues sous le nom d'algorithmes évolutionnaire), de celles qui ne manipulent qu'une solution à la fois(on parle alors des méthodes trajectoire). Les approches « trajectoire » se basant sur la recherche locale, consistent en des algorithmes ayant comme principe, de partir d'une solution initiale, obtenue de manière exacte ou par tirage aléatoire et s'en éloignent progressivement afin de réaliser une trajectoire, un parcours progressif dans l'espace des solutions. La méthode de descente, le recuit simulé, la méthode Tabou, la recherche par voisinage variable font partie de cette catégorie. Quant aux approches évolutionnaires, elles s'inspirent principalement de phénomènes naturels plus précisément de l'évolution biologique, décrivant comment les êtres vivants s'adaptent à leur environnement. Elles manipulent un ensemble de solutions simultanément, en les faisant évoluer graduellement et cela afin d'améliorer l'exploration de l'espace des configurations. Les algorithmes tels que : les algorithmes génétiques, les algorithmes par colonies de fourmis, l'optimisation par essaim particulaire, l'optimisation basée sur la biogéographie, les algorithmes à estimation de distribution, le pathrelinking (ou le chemin de liaison) se rangent dans cette catégorie.
On distingue aussi une autre catégorie de métaheuristiques : elle est dite hybride. Cette dernière agit en faisant collaborer deux métaheuristiques : telle que la méthode GRASP qui consiste en la construction d'un ensemble de solutions puis l'amélioration de celui-ci avec une recherche locale.
Enfin, on peut également différencier les métaheuristiques selon ce qu'elles possèdent comme faculté de mémoriser les informations ou non. La première catégorie mémorise les informations au fure et à mesure que leur recherche avance. Tandis que la deuxième est sans mémoire, fonctionne en aveugle et peut même revenir sur des solutions déjà examinées.
Il est aussi possible de distinguer entre le concept de mémoire à court terme et à long terme. La recherche tabou s'illustre comme étant le meilleur représentant des métaheuristiques avec mémoire, dans la catégorie sans mémoire on trouve

le recuit simulé.
Deux concepts fondamentaux sont à connaître dans le cadre des métaheuristiques à savoir l'intensification et la diversification.

- **L'intensification** consiste en l'utilisation des informations disponibles pour améliorer la pertinence de celles-ci. Du point de vue des métaheuristiques, il s'agit généralement de recherche locale. Les algorithmes de recherche locale sont maintenant souvent employés en association avec d'autres métaheuristiques plus complexes, donnant lieu à des algorithmes hybrides. On rencontre ainsi souvent l'algorithme du "simplexe" de Nelder-Mead [61], mais aussi des métaheuristiques plus complexes, comme la recherche avec tabous, qui sont parfois employées.

- **La diversification** est la recherche de nouvelles informations, afin d'augmenter la connaissance du problème. Ce sont souvent des méthodes stochastiques, et il est pour le moment difficile de dégager des idées générales, tant la diversité d'approches de cette composante des métaheuristiques est grande.

L'équilibre entre ces deux éléments est fondamentale pour aboutir à une métaheuristiques performante. Pour une meilleure compréhension des métaheuristiques, nous définissons également deux concepts importants qui sont l'optimum local et l'optimum global.

- **L'optimum global** : la meilleure solution possible à un problème.
- **L'optimum local** : il peut exister des solutions intermédiaires, qui sont également des optima locaux mais uniquement pour un sous-espace restreint de l'espace de recherche.

1.4.2.4. Les principales métaheuristiques

Nous présentons dans ce qui suit les principales métaheuristiques. Les métaheuristiques se sont imposées comme des méthodes avérées de résolution de problème d'optimisation difficile. Ces méthodes ont suscité l'intérêt de nombreux chercheurs. De ce fait, la littérature est riche de métaheuristiques très variées dont nous citons quelques unes : le recuit simulé (simulated annealing (SA)) (Aarts and Korst, 1988) [2], la recherche (tabu search (TS)) (Glover, 1989) [35], les algorithmes génétqieus (genetic algorithm (GA)) (Davis, 1991) [23], GRASP (Dyall et al., 1989) [28], la recherche à voisinage (variable neighborhood search (VNS)) (Mladenovic and Hansen, 1997)[57], les essaims particulaires (particle swarm optimization (PSO)) (Eberhart et al., 1995) [29], ant colony (Dorigo et al., 2000)[26],etc...
Nous définissons plus en détail dans ce qui suit quatre principales métaheuris-

tiques en distinguant entre les méthodes à solution unique et méthodes basées population.

1. **Les algorithmes à solution unique**
   Dans cette section, nous nous intéressons à deux des principales métaheuristiques à base de solution unique, aussi appelées méthodes à trajectoire. La spécificité de ces dernières réside dans le fait qu'elles commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Nous avons choisi de détailler deux principales méthodes de cette catégorie à savoir : la méthode du recuit simulé et la recherche tabou. D'autres méthodes à trajectoire existent telles que : la méthode de descente, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes.

   - *La recherche tabou* : la recherche tabou en anglais Tabu Search (TS) est une métaheuristique qui manipule une seule solution. Elle a été proposée par Fred Glover en 1998 [35]. Cette métaheuristique a été définie par Glover comme une métaheurisitque permettant de guider une recherche locale, pour explorer l'espace de recherche et éviter l'optimum local. Afin d'éviter les optima locaux et pour une meilleure exploration de l'espace de recherche la recherche tabou se base sur l'historique de recherche. Cette dernière utilise donc le principe de mémoire. Ainsi sa principale caractéristique est l'utilisation d'une liste tabou qui gardera en mémoire la liste de solutions déjà rencontrées. Les solutions contenues dans la liste sont alors tabou c'est à dire qu'il est interdit de les visiter.
     La méthode tabou démarre donc d'une solution initiale aléatoire ou heuristique. A chaque itération la solution actuelle est remplacée par la solution voisine non tabou même si celui-ci dégrade la fonction-objectif f. Les solutions sont taboues pour un certain nombre d'itérations. Cette liste peut être vue comme une mémoire à court terme qui garde en mémoire les solutions visitées récemment. Elle évite ainsi de revenir vers ces dernières. La taille de la liste est un élément important pour l'implémentation d'une recherche tabou. En effet en diminuant la taille de cette liste on favorise l'intensification alors que l'augmentation de la taille de liste permettra une exploration plus large de l'espace de recherche et donc aura tendance à favoriser l'exploration. Il est également possible de varier la taille de la liste au cours du processus de recherche, ceci donnera lieu à un algorithme plus robuste tel que l'algorithme tabou proposé par Battiti Tecchiolli, 1994 [6].
     Il arrive que certaines interdictions soient gênantes et donc inutiles. En effet, les interdire serait désavantageux pour la recherche. C'est dans ce cadre que l'on a pensé au mécanisme d'aspiration. Ce dernier, selon certains critères, lèvera le tabou d'un mouvement, si ce mouvement

permet d'obtenir une meilleure solution.

- *Le recuit simulé* : la méthode du recuit simulé (RS) en anglais appelé Simulated Annealing (SA) a été introduite par des chercheurs de la société [46] en 1983. Néanmoins, les origines de cette méthode remontent aux années 50. En effet cette méthode est basée sur les travaux antérieurs de Metropolis relatifs au formalisme de mécanique statistique (algorithme Metropolis) [55]. Les fondements de la méthode RS s'inspirent de la métallurgie, plus particulièrement du processus de recuit physique. En métallurgie, ce processus vise à réorganiser la structure cristallographique des métaux. Pour ce faire on alterne entre des cycles de refroidissement lent et de réchauffage (recuit), qui ont pour effet de minimiser l'énergie du matériau. Chaque température est maintenue jusqu'à ce que le matériau atteigne l'équilibre thermodynamique. Le but étant d'obtenir un état cristallisé stable du métal d'énergie minimale, tout en évitant les structures « métastables », caractéristiques des minima locaux de l'énergie.
  Cette méthode est utilisée pour la résolution des problèmes d'optimisation en appliquant la transposition suivante : la fonction coût à minimiser correspond à l'énergie du métal et les états physiques comme des configurations aléatoires. La fonction est alors minimisée en introduisant une température fictive à laquelle s'associe une fonction décroissante qui définit un schéma de refroidissement. L'algorithme démarre d'une solution initiale s générée de façon aléatoire ou par heuristique. La solution est évaluée suivant la fonction f et a pour coût (on parle également de qualité) $f(s)$. Un paramètre température T est initialisé. A chaque itération l'algorithme génère une solution $s^{'}$ aléatoirement dans le voisinage de la solution s. L'algorithme évalue la qualité de la solution $s^{'}$ et à cette étape deux choix sont possibles : accepter ou non la solution.
  - Si $(f(s^{'}) <= f(s))$ alors $s = s^{'}$
  - Sinon Si $f(s^{'}) > f(s)$
    a) r nombre aléatoire $\in [0, 1]$
    b) $\Delta(f) = (f(s^{'}) - f(s))$
    c) $P = \exp^{(-\Delta(f)/T)}$
    d) Si $r < P(T, \Delta(f))$ alors $s = s'$

  L'algorithme retient la solution $s^{'}$ si cette dernière est de performance supérieure ou égale à celle de la solution courante, dans le cas contraire $s^{'}$ est acceptée avec une probabilité. Pour cela, l'algorithme tire un nombre aléatoire r entre 0 et 1. Ensuite on calcule la probabilité : $P(T, f(s), f(s^{'}))$. Pour cela nous avons besoin de l'énergie $\Delta(f)$ et de la température T ensuite on obtient P en appliquant l'équation 1.1. La

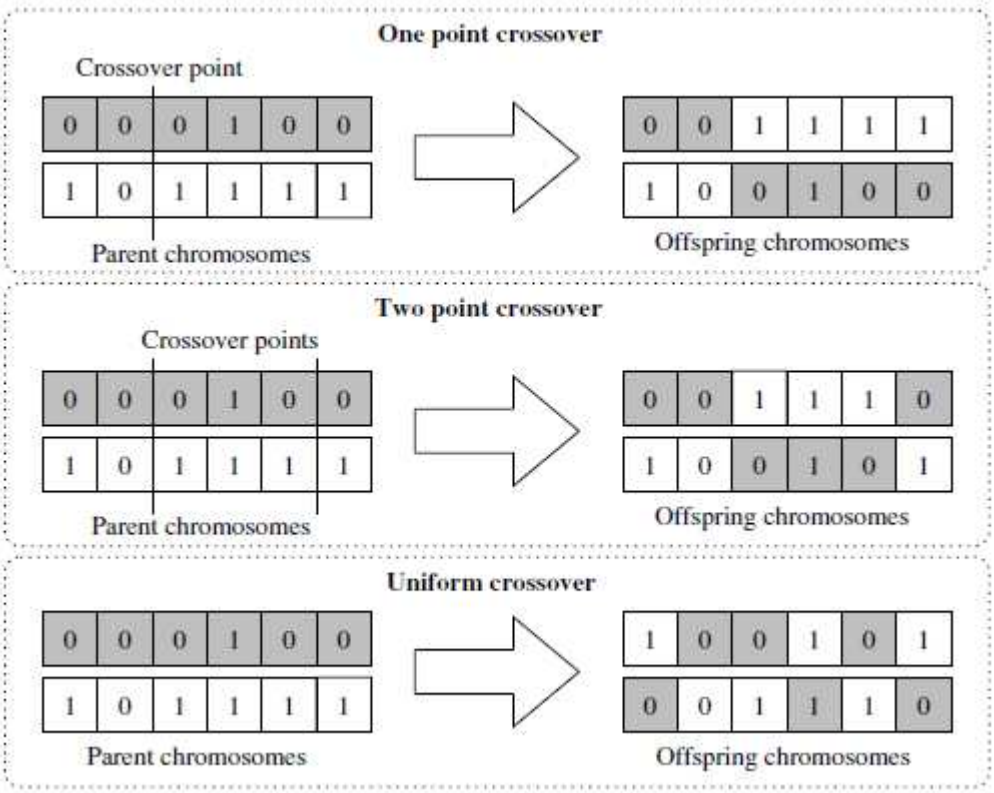
solution est acceptée si le nombre aléatoire r est supérieur à la probabilité P.

$$\exp(\frac{-\Delta(f)}{T}) \tag{1.1}$$

La température T est diminuée suivant l'évolution du processus. Ainsi, au début du processus, il existe plus de probabilité de retenir des solutions non améliorantes. De ce fait, au fure et à mesure que la température baisse la probabilité de retenir des solutions de moins bonnes qualités diminue progressivement. L'équilibre entre l'intensification et la diversification- élément important dans une métaheuristique- est assuré par le paramètre température T. En effet, la diminution progressive de T au cours du processus de recherche, tel qu'au départ la probabilité est proche de 1 autorisant ainsi la validation des solution variées, permet d'explorer au départ une zone de recherche plus vaste pour ensuite intensifier la recherche vers la fin de l'algorithme en acceptant de moins en moins les solutions de mauvaises qualités.

2. **Les algorithmes basés population**
   - *Les algorithmes génétiques* : les algorithmes génétiques (AG) que nous devons aux travaux réalisés par de John Holland (1997)[45] sont sans conteste les algorithmes évolutionnaires les plus célèbres. Ces derniers, s'inspirent de la nature plus particulièrement ils s'inspirent de la théorie de l'évolution. Pour être plus précis, ils se fondent sur les règles de la génétique qui décrivent la capacité des individus à survivre et à s'adapter à leur environnement.
     Les algorithmes génétiques manipulent un ensemble de solutions appelé population, en combinant des opérateurs pour chaque étape. Nous décrivons dans ce qui suit les différents opérateurs :
     - **Sélection** : les individus les mieux adaptés à l'environnement dits les plus forts tendent à suivre et sont par conséquent plus enclins à se reproduire. Il existe plusieurs techniques de sélection, nous citons les plus utilisées : sélection par tirage à la roulette (roulette-wheel selection), la sélection par tournoi (tournament selection), la sélection par rang (ranking selection), etc... [36, 11].
     - **Croisement** : L'opérateur de croisement ou de reproduction donnent naissance à des individus ayant hérités leur caractéristiques d'autres individus parents[13]. Cette opérateur combine alors les caractéristiques d'un ensemble d'individus parents (généralement deux) préalablement sélectionnés, et génère de nouveaux individus enfants. Il existe de nombreux opérateurs de croisement, par exemple : le croisement en un point, le croisement en n-points $(n > 2)$ et le croisement uniforme 5 (voir figure 1.1).
     - **Mutation** : Les individus créés sont mutés, c'est-à-dire que certaines

FIGURE 1.1. – Le croisement dans l'algorithme génétique



caractéristiques sont modifiées aléatoirement, selon l'opérateur de mutation. Ceci permet d'introduire de nouvelles capacités d'adaptation à l'environnement, capacités qui pourront se propager grâce aux mécanismes de sélection et de croisement[41]

- **Remplacement** : Le remplacement (ou sélection des survivants), remplace certains des parents par certains des descendants. Le plus simple est de prendre les meilleurs individus de la population, en fonction de leurs performances respectives afin de former une nouvelle population.

- *Les colonies de fourmis* : l'optimisation par colonie de fourmis (ACO : Ant Colony Optimization) est une métaheuristique basée population proposée par Dorigo et al. en 1996. Cette méthode tire son inspiration du comportement des fourmis pour trouver le plus court chemin de leur nid jusqu'à la nourriture. Les fourmis sont capables de trouver un tel chemin en raison de la capacité que possèdent les fourmis à communiquer entres elles de manières indirecte. En effet, lorsqu'elles explorent le chemin à la recherche de la nourriture, elles laissent derrière elles une traînée de substances chimiques, appelées phéromones afin de marquer certains chemins favorables qui devraient guider leurs

congénères à la source de nourriture [25]. Les fourmis qui retournent plus rapidement au nid en passant par la source de nourriture sont celles qui ont emprunté le chemin le plus court. Ainsi, la quantité de phéromones déposée sera plus importante sur ce chemin. Ce chemin aura alors plus de chances d'être emprunté. En se basant sur leur expérience collective, elles finissent par construire une solution au problème. Des travaux utilisant cette approche sont décrits dans [21].

1.5. Synthèse

Dans ce chapitre nous avons commencé en premier lieu par introduire les notions de base de l'optimisation combinatoire. Nous avons également donné un aperçu des principales méthodes de résolutions en se focalisant particulièrement sur les métaheuristiques. Ces dernières offrant une alternative intéressante pour faire face aux problèmes de l'explosion combinatoire. La recherche dans le domaine des métaheuristiques ne cesse d'avancer et de nouvelles métaheuristiques ne cessent d'être proposées.
C'est d'ailleurs dans ce contexte que s'inscrit notre démarche. En effet nous avons choisi d'exploiter cet outil puissant de résolution de problèmes complexes que sont les métaheuristiques avec pour ambition de résoudre efficacement le problème de minimisation d'interférences lors de l'affectation de fréquences dans un réseau cellulaire.

2. Le problème d'affectation de fréquences dans les réseaux cellulaires

Sommaire

2.1. Introduction

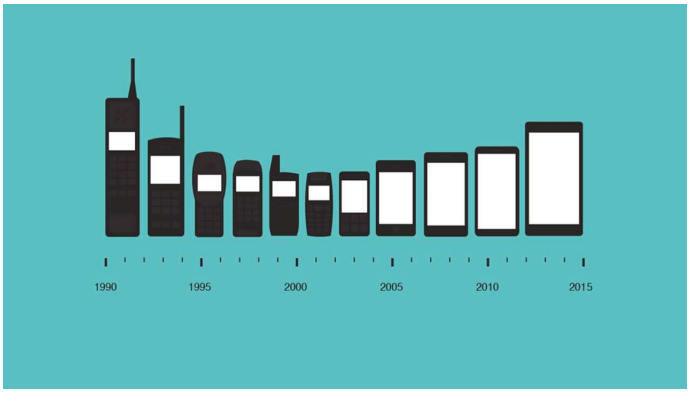
Ce chapitre est dédié à la présentation du problème que nous traitons dans cette thèse à savoir le problème de minimisation des interférences lors de l'affectation des fréquences dans un réseau cellulaire plus connu sous son nom anglais the *Minimum Interference frequency assignment problem* (MI-FAP). Nous utiliserons l'acronyme MI-FAP pour parler de la problématique traitée tout au long de ce manuscrit. Nous aborderons en premier lieu la téléphonie mobile de façon assez brève. Nous décrirons les deux concepts clés de la téléphonie mobile à savoir le concept cellulaire et la réutilisation de fréquences. Nous introduirons ensuite le problème d'affectation de fréquences et ses différentes variantes. Nous expliquerons les raisons qui font de ce problème un problème complexe et très attrayant à résoudre. Ensuite, nous ferons une description de quelques méthodes antérieures proposées pour la résolution du problème d'affectation de fréquences.

2.2. La téléphonie mobile

2.2.1. Historique

De nos jours, il est quasi impossible de ne pas disposer d'un téléphone portable, connu également sous l'appellation de téléphone cellulaire. Ce dernier est muni d'un dispositif qui permet une communication (passer ou recevoir des appels) téléphonique tout en se déplaçant à travers une vaste région géographique couverte par un réseau téléphonique. Cette communication est possible via une liaison radio. Cette opération n'est bien sûr réalisable que si l'utilisateur du téléphone est connecté à un réseau téléphonique géré par un opérateur de téléphonie mobile. La téléphonie mobile a connu bien des difficultés avant l'établissement de normes et de standards.
Les années soixante-dix ont vu l'émergence de nombreux réseaux de radiotéléphonie analogique. Connus sous l'appellation de réseaux de première génération 1G ;ces derniers ont souffert d'incompatibilités entre leurs différents standard tels que : AMPS (Advanced Mobile Phone System)lancé aux Etats-Unis, NMT (Nordic Mobile Telephone) essentiellement conçu dans les pays nordiques, TACS (Total Access Communications System), fortement utilisé en Grande Bretagne et également C450, Radiocom 2000.... C'est dans ce contexte d'incompatibilité et de réel désir de faire aboutir le système de téléphonie mobile vers de réelles normes, qu'en 1982, est créé le groupe : Groupe Spécial Mobile (GSM), par la Conférence Européenne des administrations des Postes et Télécommunications (CEPT). Ce dernier avait pour rôle d'établir des normes de communications : fixer des bandes de fréquences communes à toute l'Europe dans la bande des 900 MHz.
C'est à la fin des années 80 que le groupe GSM fixe les choix technologiques relatifs à l'usage des télécommunications mobiles : transmission numérique, multiplexage temporel des canaux radio, chiffrement des informations ainsi qu'un nouveau codage de la parole[24].

FIGURE 2.1. – L'évolution du téléphone portable



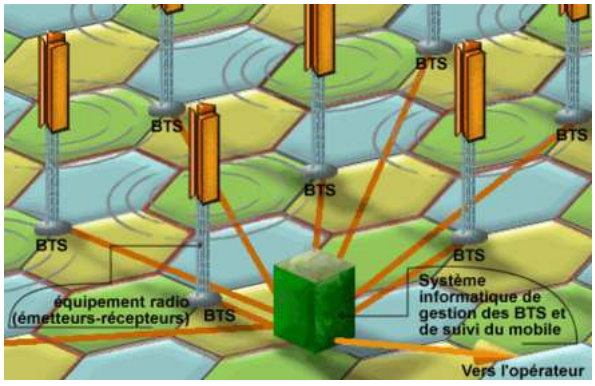
Une nouvelle norme, la DCS 1800, qui est une nouvelle adaptation de la norme

GSM exploitant les bandes 1800, voit le jour en 1990. La première communication expérimentale par le biais de cette adaptation de la norme GSM a lieu en 1991. C'est à ce moment que le sigle GSM change de nom et devient tel qu'on le connaît aujourd'hui Global System for Mobile Communications. On parle alors de réseau de $2^{ème}$ génération. Depuis lors, la téléphonie mobile n'a cessé d'évoluer pour arriver aujourd'hui aux réseaux de $5^{ème}$ génération.

2.2.2. Le concept cellulaire et la réutilisation de fréquences

La téléphonie mobile a connu et connaît toujours un succès fulgurant et les utilisateurs de réseaux téléphoniques sont en constante augmentation. Afin de répondre à cette demande, la structure du réseau doit permettre l'extension de manière assez fluide. L'espace géographique couvert par le réseau est alors divisé en cellules. La division cellulaire a pour but de garantir l'itinérance et la mobilité. En effet, le réseau doit reconnaître à tout instant la localisation de l'abonné et dans cette optique il s'agit de gestion d'itinérance ou roaming. Le réseau doit également garantir la mobilité via des procédures automatiques dites de transfert ou de hand-over, la communication doit être maintenue en cours de déplacement du mobile.
En théorie, les cellules sont représentées en hexagones. Une cellule est une zone géographique servie par une station de base (Base transceivrs station BTS). Les stations sont placées sur des sites. Un site est un espace géographique défini par ses coordonnées. Dans un même site, plusieurs stations peuvent être installées. Une station est munie de plusieurs émetteur/récepteur en anglais transceivers (TRX). Les $TRXs$ sont des équipements qui modulent le signal pour une fréquence.

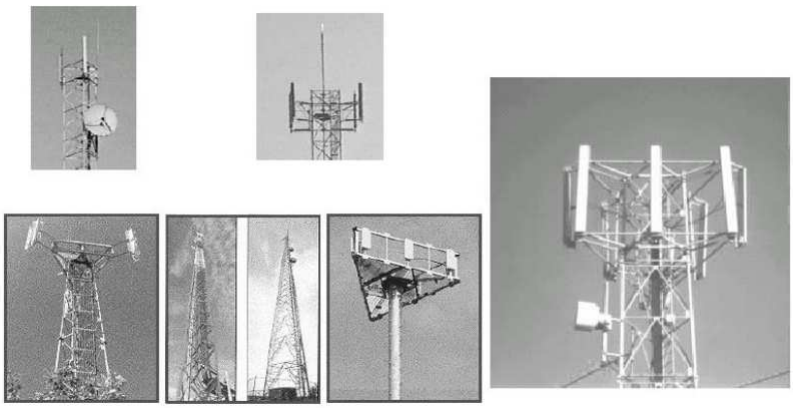
FIGURE 2.2. – Forme hexagonale d'une cellule



Le réseau mobile repose essentiellement sur l'utilisation de fréquences radio. Cependant, la ressource radio est rare et coûteuse, or le nombre d'utilisateurs est en constante augmentation. Cette situation ne laisse d'autre choix que de réutiliser les fréquences.La réutilisation des ressources radio (fréquences) constitue le

deuxième intérêt du concept cellulaire. Ce principe de réutilisation de fréquences n'est pas sans conséquences. En effet, ceci donne naissance à des interférences. De ce fait, il est primordial que les fréquences soient affectées judicieusement car la qualité de service du réseau dépend grandement de la gestion de la ressource radio et de la limitation des interférences. C'est ce qu'on appelle le problème d'affectation de fréquence. Nous définissons dans la section suivante plus en détail le problème auquel sont confrontés les ingénieurs.

FIGURE 2.3. – Station de base



2.3. Présentation du problème

Le problème d'affectation de fréquences est un problème d'optimisation combinatoire bien connu de la recherche opérationnelle. Proposé par Metzeguer[56] dans les années soixante-dix, il a depuis suscité l'intérêt de nombreux chercheurs [37, 1, 30, 49] . C'est un problème attrayant où il est question d'affecter de manière optimale les fréquences. En effet, la ressource radio est rare et les opérateurs doivent faire face à un nombre sans cesse grandissant d'utilisateurs. Par conséquent, les opérateurs disposent d'un nombre limité, défini et autorisé de fréquences et doivent affecter une fréquence à un grand nombre d' émetteurs/récepteurs (TRX) répartis sur une grande surface géographique. Étant donné que les TRX du réseau radio communiquent entre eux sur une plage de fréquences, une fréquence doit être attribuée à chacun d'eux sans, ou du moins avec le moins possible d'interférences. En effet, si des TRX proches géographiquement utilisent la même fréquence ou des fréquences ne respectant pas une certaine distance ou plus exactement ce qu'on appelle une contrainte de séparation, des interférences surviennent. L'affectation de fréquences est une généralisation du problème de coloration de graphe [37]et il a été démontré NP-complet

[1]. C'est donc un problème d'optimisation combinatoire difficile à résoudre. Différentes méthodes ont été proposées pour résoudre le FAP. Nous ferons plus loin un rappel des principales méthodes proposées.

2.3.1. Les problèmes de coloration de graphe et le FAP

En 1980 Hale [37] publie un survey où il discute de la relation entre le problème d'affectation de fréquences et le problème de coloration de graphes. Le Problème de coloriage de graphe se formule communément comme suit : considérons un graphe $G = (V, E)$ tel que V l'ensemble des sommets et E l'ensemble des arêtes. Le problème de coloration de graphe revient alors à affecter une couleur à chaque sommet. Ceci n'est pas sans contrainte. En effet il est impératif de respecter la contrainte suivante : deux nœuds adjacents (reliés par une arête), ne doivent pas avoir la même couleur. Le nombre chromatique est le nombre minimum de couleur nécessaire pour colorier ce graphe en respectant cette contrainte.
Il existe d'autres variantes telles que la T-coloration de graphe. Dans cette variante, chaque couleur est représentée par un nombre entier, chaque sommet reçoit une couleur en respectant la relation suivante : soit i,j deux sommets reliés par une arête, $|couleur_i - couleurj| \geq d_{i,j}$. C'est à dire que les couleurs doivent être affectées en respectant une distance minimale entre elles. Il y'a aussi le problème de T-coloriage ensemble (set T-coloring problem). Pour ce problème plusieurs couleurs doivent être affectées à un nœud. De plus, tout comme pour le problème de T-coloriage, une distance minimale doit être respectée entre les couleurs affectées. Le T-coloriage est un cas particulier de ce problème. La relation entre le problème d'affectation de fréquences et le problème de T-coloration est montrée dans [37]. Par analogie au problème de T-coloration ; il est alors question d'affecter les fréquences aux TRX du réseau en respectant une distance minimale entre les fréquences.

2.3.2. Les différents problèmes d'affectation de fréquences connus

Depuis son apparition le FAP a été très étudié par la communauté scientifique et bon nombre de modélisations, de variantes et de formulations de ce problème existent. Par exemple nous pouvons distinguer entre l'allocation :

- Statique : Les fréquences sont allouées en une fois à tous les $TRXs$ avant le lancement du réseau (efficace pour les réseaux chargés en demande de communication).

- Dynamique : Les fréquences sont allouées à un TRX à l'arrivée d'une

communication (utile aux réseaux peu chargés afin d'utiliser moins de fréquences que dans l'allocation statique).

- Hybride : Dans ce cas, le spectre radio est divisé en deux : un ensemble de fréquences fixes, allouées de manière fixe à autant de $TRXs$ que possible et un ensemble de fréquences dynamiques à allouer, au besoin, à un TRX sans fréquence fixe qui reçoit une communication.

Il convient de souligner également que dans la littérature, le FAP se décline sous plusieurs variantes suivant le critère à satisfaire. De ce fait plusieurs formulations ont été proposées et à ce titre, le travail publié par [1] en fait une bonne description. Aussi, nous résumons ci-dessous ces variantes :

- Affectation de fréquences d'ordre minimum (MO-FAP) : Il s'agit d'affecter différentes fréquences de sorte à minimiser leur nombre et qu'aucune interférence inacceptable n'apparaisse. En effet, lors de l'introduction des téléphones mobiles suivie de l'accroissement du nombre d'utilisateurs,un problème est survenu pour la simple raison que les fréquences se vendaient par unité et le prix de chaque unité était trop élevé.

- Affectation de fréquences de spectre minimum (MS-FAP) : Il s'agit d'affecter des fréquences qui ne permettent aucune interférence inacceptable et qui garantissent la minimisation du spectre (Span en anglais) représentant la différence entre le maximum et le minimum des fréquences utilisées. Dans cette situation, l'opérateur paye pour l'ensemble de la gamme de fréquences donc plus le spectre est large plus le coût est élevé.

- Affectation de fréquences à blocage minimum (MB-FAP) : Un blocage survient lors de la présence de certaines interférences, il s'agit, alors, de trouver une affectation partielle (et non pas la totalité du spectre) minimisant la probabilité de blocage du réseau et affectant autant de fréquences que possible (il est question alors de problème d'affectation de fréquences à service maximum Max-FAP).

- **Affectation de fréquences d'interférence minimum (MI-FAP) : Il s'agit d'affecter des fréquences aux différents $TRXs$ du réseau en limitant le nombre d'interférences. Ces interférences sont dues au non-respect de contraintes de séparations exigées entre les fréquences de $TRXs$ proches géographiquement**.

Notre travail de recherche se concentre sur l'allocation statique en effet, les jeux de données dont nous disposons préconisent une allocation statique. Notre intérêt se porte sur la dernière variante présentée à savoir la minimisation des

interférences lors de l'affectation de fréquences dans les réseaux cellulaires.

2.4. Le problème d'Affectation de fréquences : revue de la littérature

Le problème d'affectation de fréquences est un NP-difficile, il s'avère donc difficile à résoudre de manière exacte. En effet, le nombre de solutions possibles étant très important, l'énumération de chacune de ces solutions serait impossible. Les métaheuristiques offrent une bonne alternative pour résoudre ce type de problème. De nombreux travaux ont été consacrés à la résolution du MI-FAP et on retrouve dans la littérature l'emploi des deux catégories de métaheuristiques à savoir les méta-heuristiques à solution unique et les métaheuristiques à population. Dans la catégorie des métaheuristiques à population, il y a lieu de citer, en exemple, le travail proposé en 2007 [51] utilisant les algorithmes évolutionnaires et les colonies de fourmis, l'algorithme culturel [3]. Les algorithmes génétiques ont été également proposés.Ces derniers sont des méthodes adaptatives basées sur le processus d'évolution des organismes biologiques à travers des générations. L'adaptation la plus importante au FAP est donnée dans [40] où une population (ensemble de solutions pas forcément admissibles) est générée ; des mutations sont opérées aléatoirement sur quelques individus (une mutation étant le changement de la fréquence d'un TRX, chromosome) ; des individus parents sont choisis aléatoirement pour croisement et une sélection des meilleurs individus est opérée de sorte à garder une taille de population fixe. Le processus est répété jusqu'à stagnation de la population (i.e. aucune amélioration ne s'effectue sur les individus (solutions après un certain nombre d'itérations).
Certains travaux ont été dédiés aux méthodes à trajectoire et nous en décrivons quelque uns :
La recherche taboue est une recherche locale qui contrairement à une recherche locale standard autorise la sélection d'une moins bonne solution. A chaque itération la meilleure solution parmi les solutions voisines est sélectionnée (même si cette dernière n'améliore pas la solution actuelle). Afin d'éviter de revenir vers les solutions précédentes, une liste dite taboue est utilisée pour garder la trace des derniers mouvements. Cette méthode présente des inconvénients comme l'augmentation du temps de la recherche, la difficulté de choisir la bonne taille de la liste taboue et le choix du critère d'arrêt. Ces deux derniers paramètres affectant sérieusement la performance de la technique. Les différents algorithmes utilisant la recherche taboue proposés pour résoudre le problème d'affectation de fréquences diffèrent dans certains aspects tels que :la façon de générer la solution initiale, le choix de la fonction objective, la fonction de voisinage. Par exemple nous avons les deux travaux proposés par : Costa [20] en 1993, Hao et perrier [42] qui ont recours à la recherche taboue. Le premier utilise DSATUR pour générer la solution initiale alors que Hao et Perrier utilise une recherche

locale standard. En 1995, Bouju et al [12] traitent le MI-FAP en utilisant une recherche taboue qui génère une solution initiale de manière aléatoire, la fonction objective est la somme des contraintes non respectées et le voisinage est représenté par les K sommets engendrant le plus de violations. Dans Castelino et al [17], les auteurs proposent un voisinage sans restrictions. En 2003 Montemanni propose une recherche tabou avec une liste dynamique [58]. D'autres travaux dédiés à la recherche taboue pour le MI-FAP ont été publiés tels que : [43, 60], un travail propose une hybridation entre algorithme génétique et la recherche taboue [54], un travail relativement récent propose l'algorithme du path relinking en 2015 [48].
Toujours dans le cadre de méthodes à trajectoire nous avons également le recuit simulé. Le recuit simulé est une méta-heuristique probabiliste qui émule le processus de refroidissement des métaux (après chauffage).C'est une recherche locale qui permet de s'échapper d'un optimum local et cela en choisissant comme nouvelle solution à partir du voisinage, celle qui améliore la précédente ou une moins bonne solution selon une probabilité d'acceptation. Celle-ci dépend de la température qui est un paramètre dont la valeur diminue plus le nombre d'itérations effectuées augmente, ce qui représente le processus de refroidissement. En effet plus on approche de la bonne solution plus la probabilité d'accepter de mauvaises solutions diminue. Différent travaux ont proposé le recuit simulé comme méthode pour traiter le problème d'affectation de fréquences, chacun l'a développé avec ses spécificités et ses propres jeux de données et nous citons comme exemple : [8, 27].
On trouve également dans la littérature l'utilisation de systèmes multi agents (SMA). Ces derniers sont très peu adaptés au FAP. Nous citons la méthode COSEARCH [73] basée sur l'utilisation de trois agents complémentaires, chacun d'eux ayant un rôle bien défini. Ces trois agents évoluent en parallèle et coopèrent via une mémoire partagée (Adaptive Memory, AM). Pendant la recherche, l'AM regroupe l'ensemble des connaissances acquises par les trois agents : (1) l'agent SA (Searching Agent) (2) l'agent DA (Diversifying Agent) et (3) l'agent IA (Intensifying Agent). Cette méthode combine à la fois une stratégie de diversification et une stratégie d'intensification. Ainsi, de nouvelles régions sont visitées et les régions déjà visitées font l'objet d'une étude plus précise. Cette méthode parait intéressante du point de vue de la combinaison, en parallèle de plusieurs méthodes différentes. Bien qu'en termes de qualité cette méthode donne de très bons résultats, en termes de temps ceux-ci ne sont pas les meilleurs obtenus jusqu'à présent.
L'apprentissage non-supervisé est aussi une option de résolution du FAP. Ainsi, Kunz dans Kunz [47] définit le standard d'adaptation des RNs au FAP. Dans cette formulation, (1)l'état interne d'un neurone est la fréquence du TRX correspondant ; (2) des liens sont créés entre deux neurones si les $TRXs$ correspondants participent à une contrainte et (3) une fonction d'énergie est définie à partir du poids des liens entre les neurones et leur état interne. Les mises à jour usuelles

des neurones sont effectuées de telle sorte à accomplir le but d'optimisation. Comme pour les autres méthodes, le réseau peut stagner dans un optimum local probablement dû à une mauvaise définition de la fonction d'énergie (fonction d'évaluation de la solution) et du mécanisme de mise à jour du réseau.
Ont également été proposées : des hyperheuristiques parallèle en [68], d'autres méthodes [52, 4], aussi [32, 18]
Bien d'autres travaux dédiés au FAP existent, ceci n'est pas une liste exhaustive.

2.5. Formulation Mathématique du problème MI-AFP

Le problème de minimisation des interférences lors de l'affectation de fréquences (MI-FAP) est une généralisation du problème de coloration de graphe, de ce fait il est typiquement modélisé en utilisant la théorie des graphes. Nous présentons ci-dessous une modélisation simple et basique communément proposée par [58].
Conciderons F une liste de fréquences et un graphe non orienté et doublement pondéré$G(V, E, S, P)$ où :

- V : L'ensemble des sommets du graphe représentant les transmetteurs ($TRXs$) du réseau.
- E : L'ensemble des arrêtes du graphe représentent les risques d'interférences. En effet, soit une paire de $TRXs$ représentée par les sommet u,v : il existe une arrête $e_{u,v}$ de E ssi u et v sont proches géographiquement . Chaque arrête $e_{u,v}$ est pondérée par deux poids S, P.
  - S :C'est l'ensemble des séparation $s_{u,v}$ indiquant la contrainte de séparation à respecter entre les fréquences affectées aux $TRXs$ u et v.
  - P : C'est l'ensemble des pénalités p_{uv} mesurant le degré d'interférences causé par le non-respect des contraintes de séparation s_{uv} requises entre les fréquences affectées aux $TRXs$ u,v.

De ce fait, une affectation de fréquence constitue un mappage f défini de $TRXs$ vers F ($f : V- > F$) tels que V représente l'ensemble de $TRXs$ et F l'ensemble des fréquences opérationnelles. La solution au problème revient à affecter à chaque TRX du réseau une fréquence en ne causant pas d'interférences dans le meilleur des cas ou en les limitant au maximum. En pratique, ceci revient à affecter les fréquences à chaque TRX de façon à respecter les contraintes de séparation entre chaque paire de TRX contraints, autrement l'affectation serait pénalisée. En effet, une affectation de fréquences ne respectant pas une contrainte de séparation implique une pénalité p.
Considérons, comme exemple une paire de $TRXs$ (u,v) sujet à une contrainte de séparation $s_{u,v}$. Aussi considérons f_u, f_v les fréquences affectées aux $TRXs$ u et v, respectivement . Cette affectation provoquerait une pénalité $p_{u,v}$ Si $|f_u - f_v| \leq s_{u,v}$

Par conséquent, une affectation de fréquences aux différents $TRXs$ du réseau serait considérée comme optimale si elle vérifiait toutes les contraintes de séparation ou si elle minimise les pénalités causées par la violation de ces contraintes de séparation.
Afin d'évaluer la qualité d'une affectation de fréquences X, on procède au calcul du niveau d'interférences, ce dernier étant représenté par des pénalités inhérentes à l'affectation de fréquences qui ne respectent pas les contraintes de séparation exigées. Ceci est pris en compte par la fonction 3.1 ci-dessous :

$$Cost(X) = \sum_{e(i,j)\in E;|f_i - f_j|\leq s_{i,j}} p_{i,j} \quad . \tag{2.1}$$

L'objectif est alors d'affecter les fréquences en minimisant les pénalités engendrées par le non-respect des contraintes de séparation : $TRX(u,v) \in E$ pour qui $|f_u - f_v| \leq s_{u,v}$.

2.6. Synthèse

Dans ce chapitre dédié à l'état de l'art du problème d'affectation de fréquences, nous avons décrit tout d'abord de manière générale la téléphonie mobile. Pour cela, nous avons fait un bref rappel de l'histoire de la téléphonie mobile. Nous avons défini après le concept cellulaire et le principe de réutilisation de fréquences élément clé de la téléphonie cellulaire. Par la suite, nous avons abordé le problème d'affectation de fréquences. Nous avons commencé par décrire la problématique puis nous avons énoncé les principales formulations proposées. Nous avons effectué également un tour d'horizon sur les stratégies de résolutions proposées dans la littérature. Enfin, nous terminons ce chapitre en donnant la formulation mathématique du problème MI-FAP.

Deuxième partie

Contributions

3. Cadre général de recherche

Sommaire

3.1. Introduction

Ce chapitre constitue une synthèse du cadre général de ce travail de recherche. L'objectif est alors de présenter les éléments qui sont communs aux différentes méthodes que nous avons étudiées . En effet, les contributions proposées au cours de ces années de thèses sont différentes dans la mesure où elles se basent sur quatre méthodes. Même si les méthodes sont différentes certains éléments sont néanmoins partagés et il serait donc redondant de les présenter dans chacun des prochains chapitres. Voilà pourquoi nous proposons de les synthétiser dans ce chapitre. Nous présenterons en premier lieu la modélisation du problème, le codage de la solution, la fonction objectif. Nous présenterons en second lieu une description détaillée du cadre expérimental : environnement matériel, logiciel, benchmark.

3.2. Modélisation du problème

Comme nous l'avons déjà mentionné dans le chapitre 2 le MI-FAP fait partie de la classe des problèmes de coloration de graphe. De ce fait, une modélisation par graphe est possible. Nous rappelons dans cette section la modélisation présentée dans la section 2.5 du chapitre 2 et que nous avons suivie pour réaliser nos travaux de recherche. Considérons F une liste de fréquences et un graphe non orienté et doublement pondéré $G(V, E, S, P)$ où :

— V : L'ensemble des sommets du graphe. Chaque sommet représente un transmetteur (TRX) du réseau.
— E : L'ensemble des arêtes du graphe. Chaque arête représente un risque d'interférences entre les $TRXs$ qu'elle lie. En effet, soit une paire de $TRXs$ représentée par les sommet u, v : il existe une arête $e_{u,v} \in E$ ssi u et v sont proches géographiquement. Chaque arête $e_{u,v}$ est pondérée par deux poids S, P.
  — S :C'est l'ensemble des contraintes de séparation $s_{u,v}$ indiquant la contrainte de séparation à respecter entre les fréquences affectées aux $TRXs$ u et v.
  — P : C'est l'ensemble des pénalités $p_{u,v}$ mesurant le degré d'interférences causé par le non respect des contraintes de séparation $s_{u,v}$ requises entre les fréquences affectées aux $TRXs$ u, v.

De ce fait, une affectation de fréquences représente un mappage f défini de $TRXs$ vers F ($f : V- > F$) tel que V représente l'ensemble de $TRXs$ et F l'ensemble des fréquences opérationnelles. La solution au problème revient à affecter à chaque TRX du réseau une fréquence en ne causant pas d'interférences dans le meilleur des cas ou en les limitant au maximum. En pratique, ceci revient à affecter les fréquences à chaque TRX de façon à respecter les contraintes de séparation entre chaque paire de TRX contraints, autrement l'affectation serait pénalisée. En effet, une affectation de fréquences ne respectant pas une contrainte de séparation implique une pénalité p.

3.2.1. Fonction objectif

Le problème d'affectation de fréquences dans un réseau cellulaire consiste à trouver une affectation de fréquences à chaque $TRXs$ du réseau qui minimise les interférences susceptibles d'être engendrées par le non respect de contraintes de séparation requises. La fonction objectif exprimée par la formule 3.1 déjà présentée au chapitre 1 est celle que nous utiliserons dans toutes nos contributions pour évaluer les solutions manipulées par les algorithmes proposés. Cette dernière permet d'évaluer le niveau d'interférences généré par l'affectation des différentes fréquences aux ($TRXs$) du réseau en appliquant une pénalité à chaque affectation ne respectant pas une contrainte de séparation requise. La fonction est donnée par la formule suivante :

$$Cost(X) = \sum_{e(i,j) \in E; |f_i - f_j| <= s_{i,j}} p_{i,j} \quad . \tag{3.1}$$

3.2.2. Codage de la solution

Trouver le bon codage d'une solution d'un problème donné est une étape cruciale qui pourrait avoir un impact sur les résultats de l'algorithme. Une solution

au problème MI-FAP est un plan de fréquences qui satisfait toutes les contraintes de séparation requises ou de pénalités minimum. Ainsi, un plan de fréquences $X = f_{12}, f_{2NF}, f_{ij} \ldots, f_{d3}$ est modélisé sous forme d'un vecteur de d variables de décisions tel que d soit le nombre total de $TRXs$ du réseau. Chaque variable f_{ik} de la solution X correspond à la k fréquence affectée au TRX i. Les fréquences sont dans l'intervalle $[1, NF]$. Afin de mieux comprendre le codage du plan de fréquence proposé, nous illustrons ceci via un exemple présenté ci-dessous : considérons un réseau cellulaire composé de 7 TRX et 4 fréquences ($d = 7$, $NF = 4$). Les TRX sont ainsi codés de 1 à 7. Les fréquences sont codées de 1 à 4.
Une solution possible est illustrée par le schéma 3.2.2

$\Downarrow$

The candidate solution X

F_{15}	F_{23}	F_{34}	F_{41}	F_{52}	F_{63}	F_{71}

3.3. Description du cadre expérimentale

Cette section est dédiée à la présentation du cadre expérimental suivi pour les différents algorithmes. Nous commencerons par décrire l'environnement matériel et logiciel que nous avons utilisé pour réaliser ce travail. Nous présenterons ensuite le jeu de données utilisées pour valider les méthodes. Une description des différentes données sera présentée.

3.3.1. Description de l'environnement matériel et logiciel

Tous les algorithmes que nous avons développés au cours de nos travaux de recherche et présentés dans ce manuscrit sont codés en utilisant le langage C dans l'environnement Windows. Les algorithmes ont été implémentés et exécutés avec un pc personnel CORE i5, 4Go de Ram. Nos tests sont effectués sur les instances d'un benchmark public dont les caractéristiques sont décrites dans le tableau 3.1. Ces benchmarks ont été utilisés par les chercheurs [58, 59, 60, 48]. Ils sont fournis sous la forme de fichier.

3.3.2. Description des benchmarks

Dans le but de réaliser nos expérimentations, nous avons conduit nos tests sur les instances d'un benchmark public. Nous présentons quelques informations concernant ces instances [58] :

— AC-x-y : Données proposées dans [58].
— GSM-x : Ces données sont des données d'un réseau GSM comprenant x nombre de $TRXs$.

— Testx : Ces données ont été proposées par l'université de Cardiff. x correspond au nombre de $TRXs$ du réseau.
— P06-z and P06b-z : Ces données sont les données du célèbre Philadelphia problem.
— GSM2-x : Données représentant les données d'un réseau GSM tel que x correspond au nombre de $TRXs$ du réseau. Ces données sont fournies par [59].
— r1-r2-s-x-w-ms-mp : Ces données ont été générées aléatoirement et fournies par [59]. La désignation de ces instances est donnée en lettre et chacune de ces lettres correspond à une caractéristique technique du réseau exemple : la lettre x correspond au nombre de TRX. Pour plus d'informations concernant ces caractéristiques techniques il y a lieu de se référer à [59].

Nous résumons dans le 3.1 les caractéristiques des différentes instances sur lesquelles nous avons travaillé tout au long de cette thèse. La première colonne donne le nom de l'instance. La deuxième et troisième colonne nommées $TRXs$, E correspondent respectivement au nombre de $TRXs$ et au nombre de contraintes de séparation. Les deux autres colonnes à savoir p_{uv} et s_{uv} correspondent à la moyenne des pénalités et la moyenne des séparations caractérisant chaque instance.

TABLE 3.1. – Caractéristiques des instances du benchmarks

Instances	TRX	E	p_{uv}	s_{uv}
AC-45-17	45	428	1	0.29
AC-45-25	45	801	1	0.34
AC-95-9	95	781	1	0.00
AC-95-17	95	2298	1	0.15
GSM-93	93	1073	1	0.28
GSM-246	246	7611	1	0.32
Test95	95	1214	1	1.37
Test282	282	10430	1	1.38
P06-5	88	3021	1	0.58
P06-3	153	9193	1	0.59
P06b-5	88	3021	1	0.39
P06b-3	153	9193	1	0.3
GSM2-184	184	6809	$8.95 * 10^6$	0.2
GSM2-227	227	10088	$9.10 * 10^6$	0.18
GSM2-272	272	14525	$7.95 * 10^6$	0.16
1-1-50-75-30-2-50	75	835	10.81	0.26
1-2-50-75-30-4-50	75	835	11.01	0.62
1-3-50-75-30-0-50	75	835	10.97	0.00
1-4-50-75-30-2-1	75	835	1	0.25
1-5-50-75-30-2-100	75	835	21.35	0.26
1-6-50-75-30-0-1000	75	835	2068.48	0.00

3.4. Synthèse

Ce chapitre avait pour objectif de décrire l'environnement général qui a entouré notre travail. C'est un chapitre informatif. Nous y avons présenté les éléments communs aux différentes méthodes développées ainsi que l'environnement matériel et logiciel qui sera utilisé par toutes les méthodes.

4. Recherche à voisinage variable pour le problème MI-FAP

Sommaire

4.1. Introduction

Nous consacrons ce chapitre à la description des algorithmes basés sur la recherche à voisinage variable -en anglais variable neighbouhood search (VNS) que nous avons proposé pour résoudre le problème MI-FAP. Aussi, nous présentons tout d'abord les principales caractéristiques de la recherche à voisinage variable et les éléments qui gouvernent le processus de recherche de cet algorithme. Nous décrivons également en détail l'adaptation faite de cet algorithme pour traiter le MI-FAP. Nous verrons ensuite les changements que nous avons apportés à VNS et les moyens utilisés pour proposer deux variantes de VNS pour le MI-FAP. Il est à noter que ce travail a fait l'objet de publication dans une revue scientifique. Nous présenterons et analyserons également les résultats des expérimentations obtenus. Nous terminerons ce chapitre par une synthèse générale du travail consacré à VNS pour résoudre le MI-FAP.

4.2. Recherche à voisinage variable for MI-FAP

Afin de résoudre le MI-FAP, nous avons proposé d'adapter la métaheuristiques VNS. Cette dernière est une puissante métaheurisitque à solution unique. VNS

a été proposée en 1997 par [57]. Depuis lors, les chercheurs n'ont cessé de s'intéresser à cette méthode. Elle a d'ailleurs été adaptée pour résoudre différents problèmes d'optimisation difficile. La possibilité de changer de voisinage pour une meilleure exploration de l'espace de recherche et éviter les optima locaux constitue la caractéristique de cette méthode . Le schéma général du processus de recherche de VNS inclut une recherche locale (local search en anglais : LS) afin d'intensifier la recherche autour d'une solution X. Depuis sa parution en 1997, différentes variantes de VNS ont été conçues telles que : la Recherche à voisinage variable générale, Recherche à voisinage variable réduit, Recherche à voisinage variable biaisé pour résoudre avec succès une multitude de problèmes d'optimisations.Toutes ces variantes de VNS se basent sur un principe essentiel qui fait la particularité de VNS à savoir le changement de systématique de voisinage. De manière générale afin d'adapter la VNS à un problème donné, il faudra définir deux principaux éléments qui gouvernent le processus de recherche de VNS. Ces deux éléments sont : une structure de voisinage $N_k(k = 1, \ldots k = k_{max})$ tel que k_{max} représente le niveau maximum de voisinage et une recherche locale. Pour implémenter l'algorithme de VNS il est également nécessaire de définir un critère d'arrêt. Ce dernier peut être le nombre maximum d'itération ou un temps CPU maximum. L'algorithme de VNS s'applique alors comme suit : en entrée une solution initiale s, ensuite vient une phase de perturbation (shaking) qui génère au hasard une solution s' dans le voisinage de la solution s. Une recherche locale est ensuite appliquée sur s', pour obtenir un optimum local s''. Si s'' est meilleure que s, alors on valide $s{=}s''$ et on génère une nouvelle solution dans le voisinage N_1. Dans le cas contraire, la solution courante reste inchangée et on passe au voisinage suivant.
Nous présentons, ci-après, l'adaptation de l'algorithme de VNS que nous avons adopté pour le MI-FAP.

1. Structure de la solution :L'algorithme de VNS manipule une seule solution qui correspond, dans le cas du MI-FAP, à un plan de fréquences X. Un plan de fréquences est un vecteur où chaque variable correspond à une fréquence opérationnelle du réseau.
2. Structures de voisinages : VNS explore différents voisins en appliquant des fonctions de voisinage en partant d'un voisinage petit vers un voisinage plus large. Pour réaliser cette opération la VNS explore au hasard différents voisinages de plus en plus distants de la solution courante [39]. En ce qui concerne le problème MI-FAP, une solution X' se situe dans voisinage de la solution X si X' diffère de X d'une ou de plusieurs fréquences attribuées aux $TRXs$ composant la solution. Le nombre maximum de TRX ayant des fréquences différentes entre X et son voisin X' dépend de la valeur $kmax$ qui représente le nombre maximum de voisinage. Tous les algorithmes basés sur VNS développés dans ce travail de recherche utilisent les cinq voisinages suivants ($N1, N2, \ldots, N5$) :

- **Le voisinage** $N1$ **:** Ce voisinage inclut l'ensemble des solutions obtenu, dans le premier voisinage de la solution courante X, en changeant la fréquence d'un TRX ayant engendré des interférences.
- **Les voisinages** $N2$ **et** $N3$ **:** Ces deux voisinages considèrent l'ensemble des solutions dans le $2^{ème}$ et $3^{ème}$ voisinages de la solution X et qui modifient la fréquence affectée à deux (respectivement trois) $TRXs$ différents en s'assurant qu'au moins une affectation courante est pénalisée c'est à dire qu'elle cause des interférences.
- **L**e voisinage $N4$, $N5$: Le quatrième et cinquième voisinage incluent l'ensemble des solutions qui diffèrent de la solution courante X par respectivement quatre et cinq affectations de fréquences que ces dernières soient pénalisées ou non. Les voisinages $N4$, $N5$ demandant la modification de l'affectation de 4, 5 fréquences, il est nécessaire de choisir 4 (5) $TRXs$ différents de la solution courante et changer leurs fréquences. Pour réaliser ce choix nous avons choisi d'utiliser la représentation "the random Key encoding" [7] en vue d'obtenir une séquence de TRX différents. En d'autres termes, chaque fois que l'algorithme de VNS doit générer une solution voisine faisant partie du voisinage $N4$, $N5$, le processus génère d nombre aléatoires dans l'intervalle $[0, 1]$. Ensuite, ces nombres sont classés par ordre décroissant. La position de chaque nombre correspond à un TRX, la VNS modifie ensuite la fréquence assignée aux 4, 5 premières positions qui correspondent à des $TRXs$. Ce processus est explicité par l'illustration suivante 4.1.

3. La recherche locale : Nous proposons d'utiliser une recherche locale des plus basique. En effet, c'est une procédure itérative qui prend en entrée la solution X et intensifie la recherche autour de cette solution pour obtenir en sortie une solution X' de meilleure qualité.

Considérons une solution X_i de sept $TRXs$ à laquelle on applique le voisinage $N5$ de VNS. Il s'agit alors de trouver une séquence de 5 $TRXs$ différents auxquels de nouvelles fréquences doivent être affectées. Le premier vecteur de la figure 4.1 montre la solution courante X_i. Le deuxième vecteur inhérent toujours à la figure 4.1 indique la représentation du chiffrement par clés aléatoires. Ces clés sont rangées par ordre décroissant dans le troisième vecteur. En considérant que chaque position de clé correspond à un TRX nous obtenons la séquence de $TRXs$ suivante $\{2, 1, 5, 7, 3\}$. La fréquence de chacun de ces $TRXs$ sera alors modifiée afin d'obtenir la solution voisine. Nous indiquons en vert les fréquences des $TRXs$ sélectionnés dans le dernier vecteur de la figure 4.1

Solution X_i

F_{15}	F_{23}	F_{34}	F_{41}	F_{52}	F_{63}	F_{71}

Le chiffrement par clé aléatoire correspondant à la solution X_i

0.5	0.6	0.3	0.06	0.45	0.12	0.32

$\Downarrow$

Le tri du chiffrement X_i

0.6	0.5	0.45	0.32	0.3	0.12	0.06

$\Downarrow$

Les fréquences des cinq TRX sélectionnés X_i

F_{23}	F_{15}	F_{52}	F_{71}	F_{34}	F_{63}	F_{41}

FIGURE 4.1. – Une illustration de l'application du random key encoding dans le cas N_5

La procédure de VNS se décline en trois phases bien distincte. Après avoir décrit tous les éléments nécessaires à l'implémentation de VNS, nous présentons dans ce qui suit les différentes phases et nous détaillons leur adaptation à MI-FAP :

1. **L**a phase d'initialisation : cette première phase implique différents éléments nécessaires à l'implémentation de VNS. Ces éléments sont les suivants :
   - Les structures de voisinages : Nous avons définis cinq structures de voisinages qui sont simples à implémenter et particulièrement adaptées au problème que nous traitons. Ceci étant fait à chaque itération pour générer un voisin, le processus applique un opérateur qui change la fréquence de $L\ TRXs$ choisis. Par conséquent, on considère $X^{'}$ dans le k^{th} voisinage de X si $X^{'}$ diffère de X de L fréquences.
   - The stopping criterion : Pour tout algorithme il est nécessaire de définir un critère d'arrêt. Pour notre implémentation nous avons choisi le temps CPU.
   - Génération de la solution initiale X : la solution initiale est générée de manière aléatoire. Ce qui veut dire que chaque fréquences peut être affectée à chaque TRX avec la même probabilité.
2. **L**a phase principale :cette seconde phase peut être considérée comme l'épine dorsale du processus de recherche de VNS. Nous allons voir plus en détail les éléments de cette phase. Cette dernière est comme suit :
   - Mettre $k := 1$.
   - Jusqu'à ce que $k := k_{max}$ répéter ces deux phases :

- ***The shaking step*** : au cours de cette étape, VNS applique certains changements à la solution courante X en générant une solution voisine $X' \in N_k(X)$. On obtient alors : $X' = shaking(X)$.
- ***The local search step*** : au cours de cette étape, VNS applique une recherche locale avec la solution voisine X' en entrée, à la recherche d'un optimum local X''. La recherche locale tente alors d'atteindre une meilleure solution à travers un certain nombre d'itérations $maxIter$. La procédure de recherche locale utilisée est résumée de la manière suivante
  — **Etape 1** : Générer un voisin X_1 de la solution $X^{'}$.
  — **Etape 2** : Évaluer la qualité de la solution X_1 : **Si**($(Cost(X_1 < (Cost(X')$) **alors** : $X^{'} = X_1$.
  — **Etape 3** : **Si** condition d'arrêt non rencontrée aller à étape 1. **Sinon** return $X^{'}$.

3. **L**a phase d'acceptation : Lors de cette phase, VNS vérifie si la solution obtenue en sortie de la recherche locale est de meilleure qualité que la solution initiale. Dépendant de la qualité de la solution, le processus décide soit d'accepter la solution soit d'aller vers le prochain voisinage. Dans ces conditions, le processus agit comme suit :
   - Set $X^{''}$=Local Search($X^{'}$).
   - **Si** ($Cost(X^{''}) < Cost(X)$) *Alors* $X = X^{''}$, $k = 1$. **Sinon** set $k := k + 1$.

Nous tenons à préciser que tous les algorithmes basés sur VNS que nous résumons ici utilisent la même recherche locale et les mêmes structures de voisinages.

4.3. Recherche à voisinage variable itérée pour le MI-FAP

Après évaluation des résultats de l'adaptation faite de VNS sur le MI-FAP, il a été aisé de constater qu'il était nécessaire d'apporter davantage de diversification à cet algorithme pour traiter le MI-FAP particulièrement pour les instances de grandes tailles où les optima locaux sont nombreux. Pour répondre à cette demande nous avons développer une recherche à voisinage variable itérée en anglais Iterated variable neighbourhood search (It-VNS). Pour pallier le problème de manque de diversité,nous optons alors pour la perturbation continue de l'optimum local courant. Évidemment, il n'est pas simple de trouver le bon opérateur de perturbation. En effet, il faut un bon équilibre car trop de diversifications entraînerait un impact négatif sur le processus de recherche en éloignant le processus de la bonne région. Pour cela, nous employons une probabilité de

perturbation. Le pseudo code de la procédure de perturbation est donné dans l'algorithme 1.

Algorithm 1 The perturbation procedure for the MI-FAP.

Require: A probability P and a solution X.
Ensure: A solution X.
1: **for** $(i = 0\,;\, i < d\,;\, i++)$ **do**
2: $r <-$ a random number between 0 and 1 ;
3: **if** $(r < P)$ **then**
4: Replace the frequency of the i^{th} TRX by a different one randomly selected from the set of the operational frequencies.
5: **end if**
6: **end for**

Le mécanisme de perturbation illustré par l'algorithme 1 fonctionne comme une sous routine de l'algorithme de It-VNS (2). Cette sous routine a pour rôle de réinitialiser les variables de décision de la solution courante. Cette sous routine est gouvernée par un paramètre compteur $COUNT$ et une probabilité $P \in [0, 1]$. Cette routine prend en entrée une solution X et décide pour chaque variable de décision de modifier sa valeur en fonction de la probabilité P. La perturbation a lieu après $COUNT$ itération de la phase principale de VNS. L'emploi du paramètre $COUNT$ permet d'éviter au processus de perturber la solution trop tôt ou trop tard. Ceci permet donc une bonne intensification et une meilleure exploitation de la solution courante. La probabilité P contrôle l'intensité de la perturbation. En effet une probabilité trop élevée donnerait lieu à un random restart alors qu'une faible probabilité n'aurait aucun impact sur le processus de recherche. De ces deux faits, il est nécessaire de trouver la bonne valeur de P pour obtenir un algorithme efficace.
Ainsi, le bon choix des valeurs de P et $COUNT$ aura pour un impact d'assurer un bon équilibre entre l'intensification et la diversification dans It-VNS. En permettant une bonne diversification, le processus explorera d'autres régions et évitera d'être confiné dans un optimum local ou de diverger complètement des bonnes régions.
Nous rappelons que l'algorithme It-VNS suit les même phases principales de VNS présentées dans la section 4.2. La principale différence réside dans l'introduction de la procédure de perturbation. Nous présentons l'algorithme It-VNS dans l'algorithme 2

Algorithm 2 The It-VNS for the MI-FAP

Require: an MI-FAP instance, kmax, a solution X, P, COUNT.
Ensure: an improved solution Best

1: $Best < -X$, $Cpt < -0$
2: **while** (Stopping criterion is not met) **do**
3: $k < -1$;
4: **while** $(k <= kmax)$ **do**
5: $X' < -Shaking(X, K)$;
6: Apply Local Search on X' to obtain X'' ;
7: **if** $cost(X'') < cost(X)$ **then**
8: $X < -X''$; $k < -1$;
9: **else**
10: $k < -k + 1$;
11: **end if**
12: **end while**
13: $Cpt < -Cpt + 1$
14: **if** (Cpt==COUNT) **then**
15: $X < -Perturbation(X, P)$
16: $Cpt < -0$
17: **end if**
18: **end while**

4.4. Breakout VNS pour le MI-FAP

La seconde stratégie que nous avons élaborée dans le but d'améliorer les résultats de VNS pour résoudre le MI-FAP s'inspire du schéma général de la breakout local search (BLS). Introduite par [10] BLS est une variante de la recherche locale itérée. Elle manipule une seule solution. La spécificité de cette méthode consiste à alterner de manière itérative entre une descente classique pour aller à la découverte des optima locaux et une stratégie de diversification qui permet de se déplacer d'un attracteur vers un autre dans l'espace de recherche. BLS porte en effet une attention particulière à la stratégie de perturbation. Ainsi, BLS peut être appréhendée comme une recherche locale itérée avec un mécanisme de perturbation sophistiqué. Le mécanisme doit être correctement pensé pour permettre d'explorer d'autres régions et éviter que le processus ne se retrouve enfermé dans un optimum local. Le processus de BLS applique une recherche locale afin d'intensifier la recherche autour de la solution courante et applique l'opérateur de perturbation adéquat sur l'optimum local obtenu en se basant sur l'historique de recherche. La méthode BLS est caractérisée par une perturbation adaptative et de multiples opérateurs de perturbation. En effet, la perturbation est gouvernée par les résultats obtenus par LS. De ce fait, l'état de la recherche guide la procédure de perturbation dans la BLS pour déterminer combien de

variables de décision seront modifiées et comment. Ainsi, l'opérateur de perturbation appliqué en l'absence d'amélioration de la solution courante est différent de celui appliqué dans le cas où une meilleure solution a été trouvée. Dans le cas où aucune meilleure solution n'est obtenue après un certain délai alors un autre opérateur devrait être appliqué.
En étant constamment à la recherche de l'amélioration des performances de VNS pour résoudre le MI-FAP, nous avons pensé à intégrer à VNS le schéma général de BLS afin de booster le processus et améliorer les résultats. L'intégration de BLS dans VNS donne lieu à une variante de VNS que nous avons nommée Breakout Variable neighboorhoud search (BVNS). Ainsi, comme pour une descente classique dans BLS, VNS avec son schéma général agit comme une recherche locale, elle est alors un moyen d'intensifier la recherche autour de l'optimum local dans BVNS. L'idée retenue repose sur l'utilisation de la force de VNS pour intensifier la recherche autour d'une solution. L'algorithme exploite alors chaque point d'une façon plus approfondie en comparaison avec une simple descente. L'optimum local est ensuite passé comme paramètre d'entrée à la procédure de perturbation adaptative. La BVNS tente d'échapper aux optima locaux en explorant de nouvelles régions. Le pseudo code de BVNS est donné dans l'algorithme 3.
Nous constatons en observant l'algorithme 3 que ce dernier se base sur deux étapes distinctes :

1. La phase de recherche VNS : cette première phase repose intégralement sur l'application de VNS. En effet, au cours de cette phase BVNS suivra les phases basiques de l'algorithme VNS en commençant par l'initialisation jusqu'à la phase d'acceptation. De plus VNS contrôle l'état de la recherche et sert à améliorer la qualité de la solution. A cet effet, BVNS utilise un compteur Cpt. Ce dernier est mis soit à zéro dans le cas où une meilleure solution est trouvée, ou incrémentée de 1 dans l'absence d'une telle amélioration ($Cpt = Cpt + 1$)
2. La phase de perturbation : cette phase débute après la phase d'acceptation de la solution. Dans le but d'échapper à l'optimum local, la procédure de perturbation que nous avons proposée implémente une stratégie adaptative. Cette phase est caractérisée par ces trois procédures :
   (a) $Determine_Per_Size(LOpti, PreviousLOpti, Cpt)$, cette procédure servira à déterminer la taille de la perturbation.
   (b) $Determine_Per_Type(Per)$: cette procédure identifie le type de perturbation à appliquer.
   (c)$Per(LOpti, PerT, Per)$, cette procédure régit la perturbation appliquée à la solution d'entrée.

Nous décrivons dans ce qui suit les différents éléments nécessaires à l'implémentation de la stratégie adaptative de perturbation ; Nous devons implémenter

différentes procédures avec différents paramètres. Commençons par décrire les paramètres impliqués dans la procédure de perturbation :

- Best : La meilleure solution obtenue par BVNS.
- LOpti : L'optimum local atteint par VNS.
- PreviousLOpti : Le précédent optimum local déjà obtenu par VNS.
- NoImprovement : Le nombre d'itérations successives, sans amélioration de la solution Best, autorisé.
- Per : Un nombre indiquant le nombre de variables de décision dans la solution qu'il s'agit de modifier. Ce paramètre est adaptatif et dépend des performances de VNS. Sa valeur est comprise entre $[PerMin, PerMax]$.
- PerT : Correspond au type d'opérateur de perturbation sélectionné. En effet, nous préconisons trois opérateurs de perturbations.
- Cpt : Un compteur utilisé pour compter le nombre d'itérations successives sans amélioration.

L'implémentation de la phase de perturbation demande l'implémentation de trois différentes sous routines. Nous présentons ci-dessous ces différentes méthodes :

1. $Determine_Per_Size(LOpti, PreviousLOpti, Cpt)$: cette sous routine permet de trouver la valeur adéquate de Per. Le paramètre Per indique le nombre de variables de décision à modifier. Sa valeur est déterminée en prenant en compte les performances de VNS. Pour déterminer Per, une sous routine suit les instructions suivantes :
   - if ($Cpt == NoImprovment$) then $Per = PerMax$;
   - else :
     - if ($Cost(LOpti) == Cost(PreviousLOpti)$ then $Per = Per + 1$;
     - else $Per = PMin$;
2. $Determine_Per_Type(Per)$: cette sous routine servira à sélectionner un des trois opérateurs de perturbation. La valeur de $PerT$ est attribuée selon les valeurs de Per en suivant les conditions suivantes :
   - If ($Per = PerMax$)then $PerT = Strong_perturb()$;
   - Else if $Per = PerMin$ then $PerT = Small_perturb()$;
   - Else $PerT = Average_perturb()$.

   La différence entre ces trois opérateurs de perturbation réside dans la valeur de Per de plus dans la nature des variables de décisions qui seront modifiées. La procédure $Strong_perturb()$ appliquera le random key encoding pour générer une séquence PerMax variables de décision à modifier, que ces dernières soient pénalisées ou pas. La procédure $Small_perturb()$ appliquera quant à elle le random key encoding pour générer une séquence PerMin de variables de décision qui actuellement génèrent des interférences. Cette procédure se concentre sur les affectations de fréquences inappropriées. En ce qui concerne la routine $Average_perturb()$, elle utilise le random key encoding pour générer une séquence Per de TRX de telle sorte qu'une partie aura une affectation de fréquences ayant

provoqué des interférences.

Les choix multiples donnés par les opérateurs de perturbation permettent d'explorer différents points de recherches en assurant ainsi une bonne diversification.

3. $Per(LOpti, PerT, Per)$: Cette procédure aura pour rôle d'appliquer les modification nécessaires à Per variables de décision en utilisant l'opérateur PerT.

Algorithm 3 The BVNS general framework for the MI-FAP.

Require: an MI-FAP instance, kmax, a solution X, LOpti, PreviousLOpti, Per.
Ensure: an improved solution Best

```
while (Stopping criterion is not met) do
  /*The VNS phase*/
  k < −1 ;
  while (k <= k_max) do
    if cost(X) < cost(Best) then
      Best < −X ;
      Cpt < −0 ;
    else
      Cpt = Cpt + 1 ;
    end if
    X' < −Shaking(X, K) ;
    apply Local Search on X' to obtain X'' ;
    if cost(X'') < cost(X) then
      X < −X'' ; k < −1 ;
    else
      k < −k + 1 ;
    end if
  end while
  /* The adaptive perturbation phase */
  LOpti = X
  Per = DeterminePerSize(LOpti, PreviousLOpti, Cpt)
  PerT = DeterminePerType(Per).
  X = Per(LOpti, PerT, Per).
end while
```

4.5. Expérimentations et résultats numériques

Dans cette section, nous décrivons les expérimentations que nous avons effectuées. Nous reportons également les résultats obtenus des différentes méthodes basées sur VNS. Dans le but d'évaluer l'impact et l'efficacité des changements

apportés à VNS dans le but d'améliorer ses performances, nous entamerons dans un premier temps la présentation du réglage des paramètres. Ensuite dans un second temps, nous présentons les résultats obtenus par les trois méthodes sur les instances du MI-FAP. Nous comparons aussi nos résultats à ceux de l'état de l'art.

4.5.1. Réglage des paramètres

Nombreuses sont les métaheuristiques qui nécessitent un bon réglage des paramètres. Les algorithmes que nous proposons ne dérogent pas à cette exigence. Certains des paramètres sont communs aux trois méthodes alors que d'autres sont spécifiques à une méthode. Nous présentons dans ce qui suit le résumé de notre travail dédié aux réglages des paramètres.

1. Paramètre à valeur fixe
   - **Etude de** $KMax$
     - Définition. La valeur de $KMax$ représente le nombre maximum d'affectation de fréquences différentes entre une solution et sa solution voisine. Ce paramètre est utilisé par les trois méthodes. $KMax >= 2$.
     - Méthode. Nous avons effectué une série de test en utilisant la méthode VNS pour fixer la valeur adéquate de $KMax$. Nous avons testé VNS sur différentes instances de MI-FAP utilisant différentes valeurs de $KMax$. Les résultats obtenus sont résumés dans le tableau 4.1. Ce dernier correspond aux valeurs moyennes (Mean) obtenues sur 15 exécutions indépendantes de VNS.
     - Valeur. Après une série de tests nous avons fixé $KMax = 5$. Nous avons conclu suite aux résultats obtenus que la solution converge à $KMax = 5$. En effet lorsque KMax est trop petit ou trop grand, ceci a un impact négatif sur la qualité de la solution.
   - **Etude de** P
     - Définition. P est la probabilité utilisée dans It-VNS pour décider de modifier ou non une variable de décision.
     - Méthode. Nous avons effectué une série de tests en utilisant différentes valeurs de P. Le résultat est obtenu après 15 minutes pour 15 exécutions indépendantes de It-VNS.
     - Valeur. Les résultats sont illustrés par la figure 4.2. Comme on peut l'observer dans la figure, les résultats obtenus sur cinq instances différentes sont assez proches pour les trois valeurs données à P avec tout de même un léger avantage lorsque $P = 0.07$.
2. Valeurs attribuées selon les instances
   Nous présentons ci-dessous les valeurs des paramètres restant dont nous n'avons pas réussi à trouver de valeur fixe pour toutes les instances et pour

lesquels par conséquent nos expérimentations n'ont pas été concluantes. Nous avons appliqué une série de tests pour trouver les valeurs adéquates aux paramètres LSite, COUNT, PerMax, and NoImprovment. Malheureusement nos efforts ont été vains. En effet les instances du MI-FAP traitées sont de différentes tailles et il est quasi impossible de trouver des valeurs aux paramètres qui satisfassent toutes ces instances. De ce fait, nous avons décidé d'attribuer les valeurs de ces paramètres en fonction de la taille des instances.

- **Le paramètre** $LSite$
  - Définition. $LSite$ correspond au nombre d'itérations de la recherche locale dans VNS. C'est un paramètre commun aux trois méthodes.
  - Valeur. Des séries d'expérience de 25 tests chacune ont été effectuées sur différentes instances. Le critère d'arrêt des algorithmes était fixé à 20 minutes. Les résultats n'ont pas été concluants. Nous avons alors choisi une taille relative aux instances : $|V| * |NF$ tels tels que $|V|$ et $|NF|$ correspondent respectivement au nombre de TRX et au nombre de fréquences du réseau.
- **Le paramètre COUNT**
  - Définition. Ce nombre correspond au nombre de fois où la routine principale de VNS est répétée dans It-VNS avant que la solution ne soit perturbée.
  - Valeur. $COUNT = |V|$ tel que $|V|$ correspond au nombre de TRX dans l'instance.
- **Le paramètre** $PerMax$
  - Définition. $PerMax$ est un nombre qui correspond au nombre max de variables de décision qu'il est possible de modifier par l'algorithme BVNS.
  - Valeur. $PerMax = 0.15 * |V|$.
- **Le paramètre** $NoImprovment$
  - Définition. $NoImprovment$ ce nombre est utilisé par BVNS. Il indique à BVNS que cela fait $NoImprovment$ fois qu'aucune amélioration n'a été enregistrée. BVNS fera alors appel à un opérateur de perturbation dédié à ce cas.
  - Valeur. $NoImprovment = |V|$

4.5.2. Résultats et analyse

Avant de présenter les résultats obtenus, nous allons donner quelques détails sur les tests que nous avons effectués. En raison de la nature non déterministe des différents algorithmes, 20 essais ont été pris en compte pour chaque algorithme sur chaque instance. Afin de comparer équitablement les approches proposées avec certains ouvrages de référence [58, 59, 60, 48], nous avons pris comme critère d'arrêt la limite de temps CPU qui est fixé de 2400 seconde. Nous avons

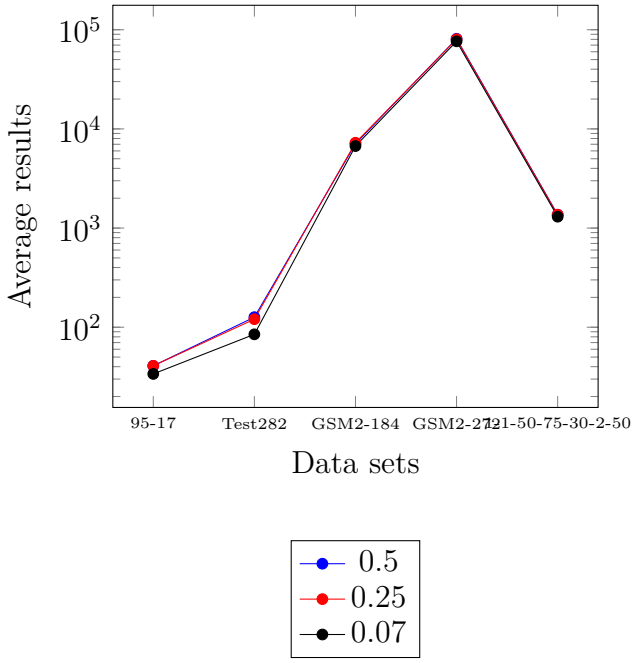


FIGURE 4.2. – Average cost results of the algorithms for some instances after 15 minutes of test and 15 runs varying the value of P

comparé nos méthodes avec les travaux mentionnés ci-dessus car les meilleurs résultats connus relatifs aux instances considérées ont d'abord été obtenus par une recherche tabou améliorée proposée par Montemanni et al [58]. De plus, en 2010, Montemanni et Smith ont proposé un algorithme efficace appelé technique de manipulation heuristique (HMT) associée à une recherche tabou [60] . Plus récemment, en 2015, Lai et al. ont obtenu de très bons résultats avec différentes stratégies basées sur un algorithme de path relinking [48]. Cependant, il convient de rappeler que les environnements de nos expériences sont différents de ceux des ouvrages de références. Par conséquent, une comparaison équitable n'est pas totalement garantie.
Les résultats obtenus pour les instances sélectionnées sont présentés dans les tableaux 4.2 à 4.7. La première colonne (Instances) indique le nom de l'instance, tandis que la seconde (F) indique le nombre de fréquences possibles du réseau. Les résultats indiqués dans la troisième colonne couvrent toutes les statistiques descriptives obtenues par les différentes méthodes, à savoir la meilleure, la moyenne et la moins bonne, ainsi que l'écart type enregistré lors de l'expérimentation. La valeur moyenne correspond à la qualité de la solution identifiée par chaque algorithme après 20 exécutions. La ligne « Best » fait référence à la meilleure valeur obtenue par chaque algorithme après 20 exécutions. Les meilleurs résultats en termes de moyenne sont indiqués en gras et la meilleure valeur est soulignée. La ligne « Worst » correspond à la solution de moins bonne

qualité, tandis que la ligne *Std* fait référence à l'écart type observé lors de l'application des méthodes proposées à l'ensemble de données.

TABLE 4.1. – Performances of the different values of KMax on five instances over 15 independent runs

Instances	KMaxValues	Average Results	Instances	KMaxV alues	Average Results
AC-95-17	3	15.6	GSM2-272	3	95760.8
	4	13		4	81760.9
	5	11.8		5	71760.5
	6	14.6		6	74760.2
Test282	3	163.3	1-1-50-75-30-2-50	3	1449.5
	4	153		4	1441.5
	5	112		5	1309.8
	6	130.5		6	1400.23
GSM2-184	3	6421			
	4	6227.3			
	5	5533.45			
	6	6332.1			

TABLE 4.2. – Results obtained by the VNS, It-VNS and BVNS algorithms with the AC-x-y instances datasets

Instances	|F|	Results	VNS	(It-VNS)	(BVNS)	Metaheuristics best results
AC-45-17	7	Best	34	32	32	32
		Average	36.75	**32**	**32**	
		Worst	39	32	32	
		Std.	1.25	0	0	
AC-45-17	9	Best	17	15	15	15
		Average	18.5	**15**	**15**	
		Worst	20	15	15	
		Std.	1.14	0	0	
AC-45-25	11	Best	35	33	33	33
		Average	36.2	**33**	**33**	
		Worst	38	33	33	
		Std.	1.26	0	0	
AC-95-9	6	Best	34	31	31	31
		Average	36	31.3	**31**	
		Worst	38	33	31	
		Std.	1.41	0.67	0	
AC-95-17	21	Best	30	17	11	10
		Average	33.25	17.1	**11.63**	
		Worst	35	18	13	
		Std.	1.86	0.30	0.8	

TABLE 4.3. – Results obtained by the VNS, It-VNS and BVNS algorithms with the GSM-x instances datasets

Instances	|F|	Results	VNS	(It-VNS)	(BVNS)	Metaheuristics best results
GSM-93	9	Best	37	38	33	32
		Average	40.25	40.2	**34.18**	
		Worst	44	42	36	
		Std.	2.52	1.03	0.89	
GSM-93	13	Best	11	8	8	7
		Average	13.8	8.8	**8.6**	
		Worst	15	10	9	
		Std.	1.31	0.63	0.51	
GSM-246	21	Best	95	84	83	79
		Average	96.76	87.72	**84.78**	
		Worst	98	90	88	
		Std.	1.03	2.32	1.71	

TABLE 4.4. – Results obtained by the VNS, It-VNS and BVNS algorithms with the Test instances datasets

Instances	|F|	Results	VNS	(It-VNS)	(BVNS)	Metaheuristics best results
Test282	61	Best	96	93	82	51
		Average	101	99.63	**84.85**	
		Worst	108	109	88	
		Std.	3.6	5.18	2.26	
Test95	36	Best	13	8	8	8
		Average	14.61	**8**	**8**	
		Worst	17	8	8	
		Std.	1.26	0.00	0.00	

TABLE 4.5. – Results obtained by the VNS, It-VNS and BVNS algorithms with the instances datasets

Instances	|F|	Results	VNS	(It-VNS)	(BVNS)	Metaheuristics best results
P06-5	11	Best	145	140	137	133
		Average	151.87	145.23	**137.26**	
		Worst	156	148	138	
		Std.	3.81	2.61	0.45	
P06-3	31	Best	123	117	115	115
		Average	127.33	117	**115.1**	
		Worst	130	117	116	
		Std.	2.19	0.00	0.31	
P06b-5	31	Best	29	26	26	25
		Average	30.15	26.2	**26.1**	
		Worst	33	27	27	
		Std.	6.56	0.41	0.31	
P06b-3 3	1	Best	113	116	112	112
		Average	121.5	121	**117**	
		Worst	130	130	120	
		Std.	4.95	4.64	3.18	

TABLE 4.6. – Results obtained by the VNS, It-VNS and BVNS algorithms with the instances GSM2 datasets

Instances	|F|	Results	VNS	(It-VNS)	(BVNS)	Metaheuristics best results
GSM2-184	39	Best	5994	6234	5898	5250
		Average	6475.4	6524.34	**6180.71**	
		Worst	6835	6786	6826	
		Std.	306.73	281.50	191.47	
GSM2-227	29	Best	71586	68586	67586	57731
		Average	77626.91	69840.82	**68721**	
		Worst	79531	75423	70566	
		Std.	2264.35	3668.68	1096.28	
GSM2-272	34	Best	68664	65568	65150	53080
		Average	71760.9	74409.9	**67888.3**	
		Worst	77806	80405	69755	
		Std.	3164.63	4466.47	1477.70	

TABLE 4.7. – Results obtained by the VNS, It-VNS and BVNS algorithms with the instances r- datasets

Instances	|F|	Results	VNS	(It-VNS)	(BVNS)	Metaheuristics best results
1-1-50-75-30-2-50	5	Best	1278	1246	1257	1242
		Average	1307.18	1290	**1268.44**	
		Worst	1344	1389	1285	
		Std.	19.25	38.33	9.12	
1-2-50-75-30-4-50	9	Best	703	672	670	665
		Average	753.61	697.69	**674.18**	
		Worst	781	766	678	
		Std.	27.78	28.77	2.89	
1-3-50-75-30-0-50	7	Best	220	195	196	194
		Average	240.61	217.69	**196.76**	
		Worst	249	232	198	
		Std.	8.64	14.40	1.09	
1-4-50-75-30-2-1	6	Best	74	74	71	70
		Average	80	75	**74.2**	
		Worst	83	77	78	
		Std.	2.53	1.24	2.25	

Dans les paragraphes suivants, nous présenterons et discuterons les résultats de chaque méthode. Nous ferons également quelques commentaires avant de tirer des conclusions préliminaires. Les résultats affichés dans les tableaux 4.2 à 4.7 montrent que les deux améliorations du système VNS, à savoir It-VNS et BVNS, ont surpassé ceux de VNS pour les différentes instances du MI-FAP. Il convient également de noter que, dans plusieurs cas, les approches améliorées proposées ont atteint le résultats de l'état de l'art, et un résultat assez proche dans d'autres cas. Examinons maintenant de plus près et analysons de manière approfondie ces résultats pour identifier les tendances et les performances des différentes méthodes appliquées pour chaque instance MI-FAP.

- **AC-xy** : le tableau 4.2 montre que, dans les quatre premières instances (AC-45-17 avec 7 et 9 fréquences, AC-45-25 et AC-95-9), instances de pe-

tite taille, les variantes de VNS améliorées (It-VNS et BVNS) surpassent les résultats de VNS. Contrairement à la méhtode VNS de base, ces variantes améliorées ont été capables d'atteindre les résultats de l'état de l'art et ceci à chaque fois. Il est à noter que ces améliorations VNS ont permis d'obtenir le meilleur résultat connu au cours des 20 tests effectués et d'enregistrer un Sdt parfait égal à 0 pour les trois premiers jeux de données ; ce qui indique que les méthodes ont prouvé leur cohérence et leur capacité à résoudre ces problèmes de petite taille. Pour l'instance AC-95-9, contrairement au BVNS, le It-VNS ne parvient pas systématiquement à identifier le best known (meilleur résultat connu). La méthode VNS de base n'a pas réussi à identifier de best known. Néanmoins VNS montre des Sdt de petite valeur qui indiquent que, dans la plupart des cas, l'optimum local correspondant au VNS n'était pas dispersé mais plutôt relativement concentré dans la même zone.

- **GSM-x instances** : Les résultats obtenus sont affichés dans le tableau 4.3. Les résultats obtenus pour les deux instances de GSM indiquent que, dans la plupart des cas, les méthodes It-VNS et BVNS sont plus performantes que VNS de base. On note également, que les résultats de It-VNS et de VNS sont similaires en ce qui concerne l'instance GSM-93 (9 fréquences). On constate aussi et toujours pour cette instance que VNS produit un résultat légèrement supérieur à celui de It-VNS en termes de solution Best. Cependant les résultats produits en moyenne par les deux méthodes restent assez proches. De plus, la valeur du std.de l'It-VNS est inférieure à celle du VNS et le moins bon optimum local est atteint par VNS. Par conséquent, contrairement au VNS, le It-VNS peut échapper à certaines régions. On peut en outre noter que les meilleures solutions identifiées par le BVNS pour les deux instances du GSM-93 sont très proches des résultats de l'état de l'art et ne diffèrent pas trop du résultat le plus connu concernant l'instance GSM-246. Les meilleurs résultats enregistrés par BVNS et It-VNS en ce qui concerne le GSM-246 sont comparables, mais en moyenne, le BVNS a produit des résultats légèrement supérieurs. Les valeurs de Sdt sont de petites valeurs dans les deux cas. On peut observer que les variantes de VNS proposées génèrent de meilleurs résultats que la méthode VNS de base dans le cas de la seconde instance GSM-93 (13). Ces variantes ont atteint des résultats très proches du best known. Même en termes de valeur moyenne, les méthodes ont presque permis d'obtenir le résultat le plus connu au cours des 20 essais. Le $Sdt.$ pour les trois méthodes est proche de zéro, démontrant ainsi la stabilité des méthodes.
- **Instance Test 95 et Test 282** : Le tableau 4.4 présente les résultats obtenus sur les instances Test par l'utilisation des trois méthodes. Concernant l'instance Test 95, on peut constater que lors de chaque exécution, l'It-VNS et le BVNS obtiennent toutes deux de bons résultats. Les résultats atteints sont ceux trouvés dans les travaux de références. Les méthodes

It-VNS et BVNS enregistrent ainsi un Std parfait, alors que VNS ne réussit pas à atteindre des résultats aussi bons. Dans le cas du test 282, les résultats de It-VNS et BVNS sont assez proches, avec un léger avantage pour BVNS. Toutefois, les deux algorithmes ont produit de meilleurs résultats que VNS.

- **Instances P06-z et P06b-z** : On observe selon les résultats présentés dans le tableau 4.5 que la méthode BVNS a donné de très bon résultats avec l'instance P06b-3. En moyenne, les résultats de VNS et de It-VNS sont assez proches pour l'instance P06b-3. Même si, VNS atteint un optimum local très proche du best known, la moyenne enregistrée montre que, de manière générale, la méthode tend à atteindre un optimum local relativement mauvais par rapport à l'état de l'art. Dans les trois autres cas, le tableau 5 montre que BVNS atteint les résultats de l'état de l'art pour l'instance P06-3 et des résultats assez proches pour les deux instances restantes. En moyenne, les méthodes It-VNS et BVNS ont toutes deux obtenu de meilleurs résultats que VNS.
- **Les instances GSM2-x** : Les résultats présentés dans le tableau 4.6 indiquent que BVNS a produit de meilleurs résultats, pour l'instance (GSM2-184), par rapport aux deux autres méthodes. On peut également noter que l'écart entre les résultats de BVNS et ceux de It-VNS n'est guère significatif. Le $std.$ enregistré pour les différentes instances GSM2 est relativement petit. Néanmoins, il apparaît que le $Std.$ de BVNS était plus petit que ceux obtenus par les autres méthodes. On peut donc en conclure que, compte tenu du fait que les résultats des différentes méthodes en termes de valeur moyenne étaient proches les uns des autres, BVNS s'est avéré plus robuste dans ces cas.

  Comme on peut le constater, les résultats obtenus par les méthodes proposées dans les instances GSM2 montrent que toutes les méthodes n'ont pas abouti aux meilleurs résultats connus et ont convergé vers un optimum local inadéquat. En effet, ces instances sont difficiles et les divers éléments qui ont été incorporés dans le schéma de base de l'algorithme VNS semblent insuffisants. En effet, alors que le schéma de base de VNS a montré sa limite en raison du manque de diversité, les schémas de perturbation de même que toutes les autres options proposées ont tendance à agir comme un redémarrage multiple aléatoire. Ainsi, au lieu d'apporter suffisamment de diversité pour réorienter la résolution vers de meilleurs résultats, où tout au moins vers un optimum local proche, toutes les méthodes conduisent systématiquement à un optimum local de moins bonne qualité ou distant par rapport aux résultats de l'état de l'art. De manière générale, étant donné que les problèmes d'optimisation combinatoire de grande taille se caractérisent par un grand nombre d'optima locaux, il n'est pas facile de diriger la recherche vers le meilleur.
- **L'instance r1-r2-sxw-ms-mp** : les résultats présentés dans le tableau 4.7

montrent que, pour l'instance 1-1-50-75-30-2-50, la BVNS a produit de meilleurs résultats que les autres méthodes en termes de valeur moyenne, mais It-VNS la dépasse en atteignant un meilleur optimum local. Néanmoins, la valeur de l'Std de It-VNS est supérieure à celle de BVNS. De plus, le $Std.$ BVNS est relativement petit. Comme on peut l'observer la BVNS a obtenu de meilleurs résultats en terme de valeur moyenne, nous pouvons donc conclure que la méthode BVSN est plus stable que la méthode It-VNS. Les deux méthodes améliorées sont plus performantes que VNS dans ce cas là. Pour l'instance 1-4-50-75-30-2-1, BVNS a obtenu de meilleurs résultats que les autres méthodes. On peut remarquer que les résultats obtenus avec la méthode BVNS sont très proches des résultats les plus connus obtenus en ce qui concerne l'instance 1-4-50-75-30-2-1. La variante BVNS a obtenu dans ces cas de meilleurs résultats que VNS et It-VNS. On peut également noter que, dans presque tous les cas, l'écart entre la qualité de la meilleure solution et la solution moyenne était généralement légèrement plus grand dans le cas de la méthode VNS.

Comme nous l'avons mentionné précédemment, l'algorithme VNS classique a convergé vers des optima locaux et a donc donné des résultats relativement mauvais. Cependant, les stratégies de perturbations que nous avons adopté pour améliorer VNS ont abouti à de meilleurs résultats. En effet,on constate aisément l'amélioration des résultats dans presque tous les cas. Ainsi, l'impact positif des changements apportés à VNS est confirmé. Les variantes améliorées que nous avons conçues, ont permis d'atteindre les performances des travaux de référence pour presque toutes les instances testées. Dans certains cas, les best known ont été atteints, parfois avec un $Std.$ parfait, prouvant ainsi leur robustesse. Alors que dans d'autres cas les résultats sont très proches du best known ; ceux obtenus montrent clairement que ces méthodes assurent un bon équilibre entre l'intensification et la diversification, base de toute métaheuristique. Nous avons donc réussi à proposer des approches améliorées de VNS qui garantissent un bon équilibre entre intensification et diversification. La méthode It-VNS est renforcée par une stratégie de perturbation qui lui permet d'éviter dans certains cas des optima locaux et d'obtenir de meilleurs résultats. Le schéma sophistiqué de perturbation proposé dans la méthode BVNS permet une large et meilleure exploration de l'espace de recherche.

4.5.3. Analyse statistique

Nous avons voulu vérifier si les différences entre les solutions trouvées par les trois méthodes étaient significatives. A cet effet nous effectuons le test non paramétrique de Friedman pour déterminer si les algorithmes sont statistiquement différents. Le test de Friedman est utilisé pour classer les performances des algorithmes pour chaque jeu de données et vérifier si la moyenne mesurée des

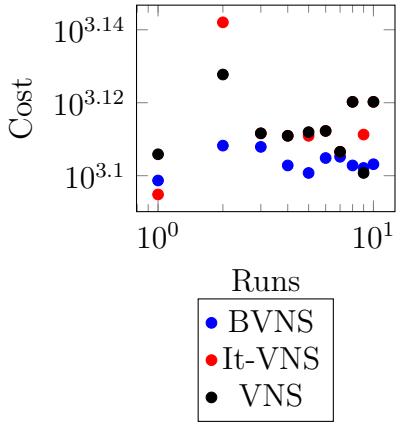


FIGURE 4.3. – Details of experiments on the instance 1-1-50-75-30-2-50

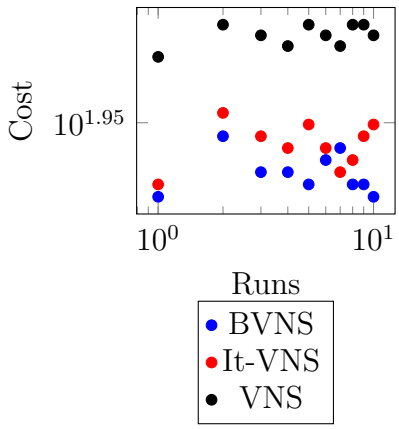


FIGURE 4.4. – Details of experiments on the instance GSM-246

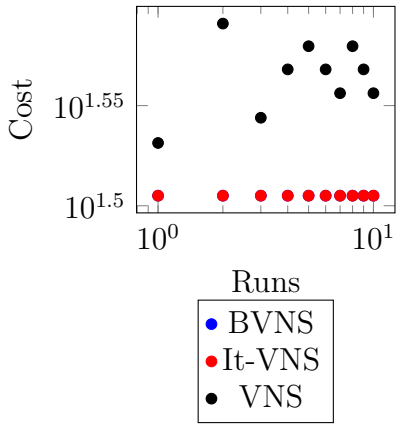


FIGURE 4.5. – Details of experiments on the instance AC45-17(7)

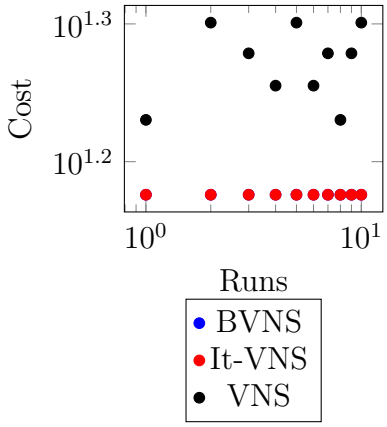


FIGURE 4.6. – Details of experiments on the instance AC45-17(9)

rangs R_j sont significativement différents du rang moyen[38]. Formellement, la statistique est définie par la formule suivante :

$$x^2 = \frac{12 * I}{|A| * (|A| - 1)} * [\sum_{j=0}^{|A|} R_J^2 - \frac{|A| * (|A| + 1)^2}{4}] \tag{4.1}$$

où A fait référence à un ensemble donné d'algorithmes alors que I représente un ensemble de données. Nous avons effectué le test de Friedman sur les jeux de données afin de détecter toute différence entre les trois algorithmes proposés. Nous avons effectué le test avec le R logiciel sous le niveau de signification alpha = 0,05.

Les résultats du test sont les suivants : Friedman chi-carré = 40,091, degré de liberté = 2, valeur p = 1,97e-09. Comme on peut le constater, le test montre la signification (valeur p <0,05), de sorte que l'hypothèse nulle peut être rejetée

car la valeur p est inférieure à 0,05. Les résultats sont donc différents. L'hypothèse nulle du test de Friedman étant rejetée, il est nécessaire de déterminer quels algorithmes diffèrent les uns des autres. À cette fin, nous avons effectué un test post-hoc de Nemenyi afin de comparer différentes paires d'algorithmes. Les résultats du test post-hoc de Nemenyi sont résumés dans le tableau 4.8. Au vu de ces résultats, nous pouvons conclure que les algorithmes comparés sont statistiquement différents. En effet, on peut voir que le BVNS diffère considérablement (p <0,05) du VNS et de It-VNS et que ce dernier diffère de manière significative du VNS (p <0,05). Il est donc évident que les modifications apportées à l'algorithme VNS par le biais de ces deux variantes, à savoir les It-VNS et les BVNS, ont eu un impact réel sur les performances de l'algorithme.

TABLE 4.8. – Nemenyi post hoc tests

Méthodes	VNS	It-VNS
It-VNS	0.0056	–
BVNS	$1.8e^{-09}$	0.0056

4.6. Synthèse

Le présent travail avait pour objet de résoudre efficacement le MI-FAP dans un réseau radio. À cet égard, nous avons proposé de résoudre le MI-FAP en adaptant d'algorithme VNS. Comme beaucoup d'autres chercheurs, nous avons privilégié cette méthode car elle repose sur un changement systématique de voisinage. Nous étions très désireux alors d'explorer en profondeur ses spécificités et d'étudier sa capacité à résoudre le MI-FAP. De plus, proposer une amélioration efficace de l'algorithme VNS pour répondre au MI-FAP constituait un défi intéressant. Nous avons donc proposé deux améliorations de VNS destinées à améliorer ses performances. Ces améliorations impliquaient l'utilisation de certaines fonctionnalités relatives à d'autres méta-heuristiques uniques, telles que la recherche locale itérée et la Breakout local search. Nos expériences ont montré que les améliorations proposées de VNS donnent de meilleurs résultats que la VNS de base.
Comme éventuelles perspectives à ce travail, il serait intéressant de proposer d'autres stratégies de perturbation et de les examiner de manière plus approfondie. Il est impératif également de centrer nos travaux futurs sur l'identification et l'élimination des limitations restantes qui empêchent les algorithmes proposés d'atteindre les résultats obtenus par d'autres chercheurs.

5. Recherche Harmonique pour le MI-FAP

Sommaire

5.1. Introduction

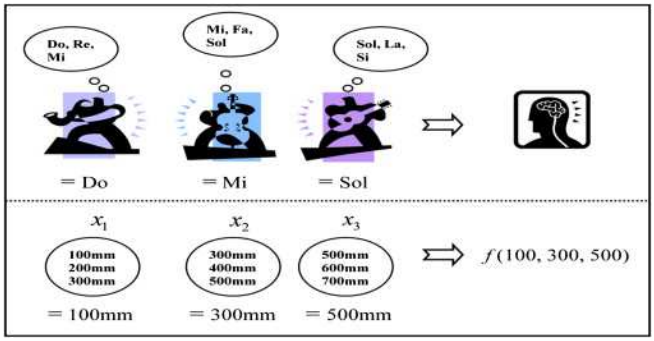
Dans ce chapitre, nous présentons une adaptation de Harmony Search (HS) pour résoudre le problème MI-FAP. Les résultats obtenus par l'adaptation de l'algorithme classique Harmony Search ne sont pas satisfaisants. En effet, nous avons effectué des tests sur des données de différentes tailles. Les résultats expérimentaux montrent que la recherche d'harmonique conventionnelle souffre de convergence prématurée et reste donc bloquée dans les optima locaux. Même lorsque le processus réussit à s'échapper de l'optimum local, il le fait après une longue période. Cela rend le processus très lent. En raison de ces résultats peu convaincants, nous avons tenté d'améliorer les performances de l'algorithme Harmony Search.
Pour atteindre cela, nous proposons dans un premier temps quelques petits changements que nous apportons à l'algorithme de recherche Harmonique d'origine et une hybridation avec une recherche locale. Nous proposons également d'exploiter le principe d'apprentissage basé sur l'opposition (OPBL). Dans cette optique, deux stratégies sont mises en œuvre pour tenter d'améliorer les performances de l'algorithme de recherche d'harmonie classique.

Nous essayerons également une autre stratégie pour améliorer les résultats de HS. Cette dernière se base sur une approche collaborative et combinera HS à une évolution différentielle. La méthode sera aussi renforcée par d'autres opérateurs que nous décrirons au cours de ce chapitre.
Pour cela nous commencerons par présenter la méthode recherche harmonique. Ensuite nous décrirons les deux méthodes que nous proposons. On présentera également les résultats obtenus après les différentes expérimentations.
Nous terminerons ce chapitre par une synthèse des différents travaux réalisés dans le cadre de l'étude de Harmony search pour le MI-FAP.

5.2. La recherche Harmonique pour le MI-FAP

La recherche harmonique ou en anglais Harmony Search (HS) est une métaheuristiques relativement récente. Elle est introduite en 2001 par[34] C'est une méthode d'optimisation globale qui a été utilisée sur de nombreux problèmes d'optimisation.La recherche harmonique est une métaheuristique basée population inspirée du processus d'improvisation suivie par un orchestre à la recherche d'une partition musicale avec une parfaite harmonie. Afin de produire une musique harmonieuse et agréable à entendre, les membres de l'orchestre ajustent plusieurs fois les tons de leur instruments.
Les concepteurs de la méthode HS ont formalisé ce processus d'improvisation en une méthode d'optimisation. Ainsi, l'analogie entre le processus d'improvisation suivi par les musiciens pour obtenir une harmonie musicale parfaite et le processus d'optimisation se fait comme le montre la figure 5.1.

FIGURE 5.1. – L'analogie recherche d'harmonie musicale parfaite et optimisation



Les différents instruments de musiques (saxophoniste, contrebassiste, guitariste) utilisés par les musiciens correspondent aux variables de décisions $(x1, x2 \ldots, x3)$. Chaque instrument va jouer différentes notes : $(saxophone = (Do, Ré, Mi)$; $contrebasse = (Mi, Fa, Sol)$ et $(guitare = (Sol, La, Si))$, chacune de ces notes correspond aux valeurs possibles que peuvent prendre les variables de décision $(x1 = (100, 200, 300)$; $x2 = (300, 400, 500)$ et $(x3 = 500, 600, 700))$. Ainsi, si le saxophoniste joue la note

Do, la contrebasse Mi et le guitariste Sol en même temps, la combinaison de ces notes forme une nouvelle harmonie (Do, Mi, Sol). Si cette nouvelle harmonie présente une meilleure norme esthétique que les harmonies existantes, elle est alors mémorisée. Par analogie la nouvelle solution ($100mm, 300mm, 500mm$) générée par le processus d'optimisation est conservée si elle est de meilleures qualité que les autres solutions.
L'algorithme de la recherche harmonique est scindée en cinq étapes. Nous résumons ces différences en établissant la correspondance avec le problème que nous traitons à savoir MI-FAP. Les étapes se résument alors comme suit :

- **Étape 1** : Cette première étape est une étape d'initialisation. Les différents paramètres gouvernant l'algorithme sont alors initialisés. Nous énumérons ci-dessous les différents paramètres :
  - La taille de la mémoire harmonique ou en anglais the harmony memory size (HMS) qui représente le nombre d'harmonies sauvegardées dans la mémoire des harmonies en anglais Harmony memory (HM). Chaque harmonie enregistrée dans HM est une solution possible et correspond à un plan de fréquences. La mémoire harmonique est la population que manipule l'algorithme HS. Elle représentée sous forme de matrice $HMS \times d$ tel que : (HMS) représente le nombre de colonnes, chaque colonne correspond à une harmonie donc une solution possible. Pour rappel d correspond au nombre de TRX du réseau.

$$HM = \begin{bmatrix} f_1^0 & \dots & f_d^0 \\ f_1^1 & \dots & f_d^1 \\ \dots & \dots & \dots \\ f_1^{HMS} & \dots & f_d^{HMS} \end{bmatrix} \tag{5.1}$$

  - Le taux de considération de la mémoire harmonique ou en anglais the harmony memory consideration rate : $(HMCR)$, $HMCR \in [0, 1]$.
  - Le taux de réglage de tons ou en anglais the pitch adjusting rate (PAR), $PAR \in [0, 1]$.

  Les deux paramètres $HMCR$ et PAR sont les paramètres qui régissent le processus d'improvisation. Ils sont alors considérés comme paramètres gouvernant les processus de recherche locale et globale de HS.
- **Étape 2** : Au cours de cette étape, l'algorithme débute par une initialisation de HM :pour cette initialisation l'algorithme génère de manière aléatoire l'ensemble des harmonies, s'ensuit l'évaluation des ces dernières en utilisant la fonction objective. Enfin chaque harmonie est enregistrée dans HM.
- **Etape 3** : L'improvisation d'une nouvelle harmonie (solution) : dans le but d'improviser un nouvelle harmonie (plan de fréquences) $H = (f_1, f_2, f_i \dots f_d)$ l'algorithme HS préconise trois choix : (1) memory consideration, (2) random selection, (3) pitch adjustment.
  Pour l'improvisation de nouvelles harmonies (Plan de fréquences) $H =$

$(f_1, f_2, f_i \dots f_d)$ trois choix sont possibles :

1. La considération de la mémoire harmonique en anglais memory consideration,
2. La sélection arbitraire en anglais random selection,
3. Le réglage du ton en anglais pitch adjustment.

Le premier choix à savoir la considération de la mémoire harmonique $HMCR$, c'est la probabilité d'affecter la $i^{ème}$ variable de décision de la $i^{ème}$ colonne dans HM à la $i^{ème}$ variable de décision de la nouvelle harmonie H créée. Sinon, la $i^{ème}$ variable de décision de H est choisie aléatoirement avec une probabilité $1 - HMCR$. Lorsque la valeur de la $i^{ème}$ variable de décision de la nouvelle harmonie H est choisie de HM, l'algorithme utilise alors PAR pour décider d'un possible ajustement. En effet suivant la probabilité PAR l'algorithme décidera d'ajuster la valeur de la variable de décision ou pas. Pour réaliser cet ajustement nous utilisons une stratégie simple et basique qui consiste en l'incrémentation par 1 ou avec une probabilité égale la soustraction 1 à l'actuelle valeur.
Pour rappel les valeurs des variables de décision correspondent à des fréquences $f \in [1, NF]$. De ce fait, dans le cas où la valeur de l'ajustement soit hors intervalle deux choix sont possibles :

$$f_i = \begin{cases} NF & \text{si } f_i < 1 \\ 1 & \text{si} f_i > NF \end{cases} \tag{5.2}$$

- **Étape 4** : Au cours de cette étape l'algorithme met à jour la mémoire harmonique dans le cas où une meilleure solution est trouvée. Pour cela chaque nouvelle harmonie est évaluée et sauvegardée dans HM si elle est meilleure que la pire solution existant dans HM.
- **Étape 5** : Répéter les étapes 3 et 4 tant que le critère d'arrêt n'a pas été rencontré.

5.3. Multiple-Harmony Search pour le MI-FAP

La première partie de ce travail a été consacré à l'adaptation de la recherche harmonique au problème MI-FAP. S'en est suivie une phase d'amélioration des performances de HS pour résoudre le MI-FAP. L'algorithme classique HS démarre par la génération des harmonies, puis à chaque génération une nouvelle harmonie est générée, évaluée et sauvegardée dans HM si elle est de meilleure qualité que la moins bonne solution dans HM. Nous avons apporté quelques changements au processus de création d'harmonies. Ainsi nous proposons de générer à chaque itération N_{new} nouvelles harmonies au lieu d'une seule, puis évaluer leur qualité. L'étape suivante consiste à choisir les HMS meilleures harmonies parmi les nouvelles harmonies et celles déjà dans HM. Suite à quoi, la mémoire HM

est mise à jour. Ce processus est répété jusqu'à la rencontre du critère d'arrêt. Ce processus permet une meilleure étude de l'information contenue dans chaque HM. En effet la création de plusieurs harmonies de HM offre une meilleure exploitation et apporte de la diversité par les nombreuses solutions. L'algorithme obtenu est baptisé MHS pour multiple-harmony search.

5.4. Oppsition based harmony search pour le MI-FAP

Toujours dans le cadre de l'étude l'algorithme de recherche harmonique pour résoudre le MI-FAP, nous avons étudié la possibilité de combiner deux différentes méthodes afin d'accélérer et améliorer le processus de recherche. Nos investigations nous ont mené à tenter une amélioration par l'ajout d' une recherche locale. Nous nous sommes également intéressés à la possibilité d'ajouter le concept de l'information opposée inspirée des machines learning, plus précisément the Opposition Based learningé (OPBL). Nous détaillerons plus loin ce concept. Nous proposons donc un algorithme de recherche harmonique, en retenant la variante MHS que nous avons proposée en section5.3, auquel vient s'ajouter une recherche locale et l'emploie de l'OPBL. Le processus de recherche démarre avec MHS, HM est alors générée de manière aléatoire puis les différentes harmonies sont évaluées. La meilleure solution $BestSol$ est sauvegardée, puis N_{new} nouvelles harmonies sont créées et évaluées. Après cela, le processus sauvegarde les meilleures HMS harmonies pour la prochaine génération.
Le processus doit ensuite vérifier si une meilleure solution que $BestSol$ a été trouvée et sauvegardée dans HM. Pour cela, l'algorithme exécute les étapes suivantes :

- Trier les harmonies dans HM de la meilleure à la moins bonne.
- *If* $(Cost(HM^0) < Cost(BestSol))$ ***Then*** $BestSol = (f_1^0, f_2^0, f_i^0 \dots f_d^0)$.
- *Else if* aucune meilleure solution trouvée ***Then*** Appel Local Search (LS).

La recherche locale (LS) est appelée dans le cas où MHS ne parvient pas à améliorer l'actuelle meilleure solution.
La recherche locale que nous utilisons dans Opp-HS est une procédure itérative qui démarre d'un solution initiale *X* prise de HM. La procédure tente alors d'améliorer cette solution, pour cela une variable de décision $f_{(i)}$ de la solution *X* est sélectionnée aléatoirement. Ensuite, elle est remplacée par une nouvelle fréquence $f_{(j)}$ choisie aléatoirement. Une solution *X'* est alors obtenue, si $Cost(X') < Cost(X)$ alors $X = X'$. Le processus est répété un certains nombre d'itération.
L'emploi de la recherche locale assure une meilleure intensification et accélère le processus de convergence. Aussi, l'introduction de LS a pour effet d'améliorer les résultats mais en contre partie guide la recherche vers un optimum local. De plus l'amélioration n'est pas toujours garantie. Il était évident que le processus

souffrirait du manque de diversité. Sur la base de ce constat, nous est venue l'idée d'exploiter l'information qui se trouve dans l'opposé des solutions de HM. L'algorithme agit alors comme suit : lorsque la recherche locale ne parvient pas à améliorer l'actuel best, la procédure OPBL est appelée.
L'apprentissage basé sur l'opposé ou en anglais the Opposition-based learning (OPBL) est inspiré des machines learning. Il a été proposé en 2005 par Tizhoos [69]. L'idée que propose Tizhoosh est fondée sur l'exploitation des potentielles informations contenues dans une solution et son opposé. L'OPBL améliore la capacité d'exploration de la recherche globale et permet également d'éviter une convergence prématurée. L OPBL est un concept qui a suscité l'intérêt de nombreux chercheurs et a fait l'objet de plusieurs travaux. Il a été utilisé dans le but d'améliorer les résultats d'algorithmes bien connus tels que : l'évolution différentielle [65], l'algorithme d'optimisation par essaim particulaire[72] et l'optimisation basée sur la biogéographie [31].
Pour trouver l'opposé d'une solution, l'algorithme remplace la variable de décision d'une solution par son opposé. Dans notre cas les variables de décisions sont les fréquences attribuées aux $TRXs$ donc l'opposé d'une fréquences f_i d'une solution X est définie comme suit : Considérons $f_i \in [1, NF]$, nous avons alors l'opposé f'_i de f_i ainsi :

$$f'_i = 1 + NF - f_i \tag{5.3}$$

La méthode nommée Opp-HS agit alors comme suit :

- Appel de la méthode (MHS).
- Tri des harmonies contenues dans HM de la meilleure à la moins bonne.
- *If* $(Cost(HM^0) < Cost(BestSol))$ ***Then*** $BestSol = HM^0$.
- *Else if* Pas de meilleur best trouvé ***Then*** Appel Local Search (LS).
  - *If* (LS) n'améliore pas le best ***Then*** remplacer chaque solution de HM par sa solution opposée en utilisant la formule 5.3.

5.5. Iterated HS-DE pour le MI-FAP

Dans la méthode HS classique, nous pouvons considérer que la tâche d'intensification est en grande partie remplie par la création de solution basée sur la mémoire et qui utilise le paramètre $HMCR$, alors que celle de diversification est exécutée par la création basée sur l'ajustement utilisant le paramètre PAR. Or, nous avons pu constater, à travers l'analyse conduite dans l'application de HS sur le problème MI-FAP, que la création par ajustement, sensée agir comme un opérateur de dispersion au sein de la population, n'a pas de grande influence sur les résultats conduisant souvent à des solutions sous-optimales. Nous avons alors pensé à exploiter une approche collaborative qui combinera HS et l'évolution différentielle. Nous proposons également de renforcer cette collaboration par un restart adaptavie des solutions. Ce nouvel algorithme nommé iterated HSDE (i-

HSDE) devrait permettre d'améliorer les résultats. Notre objectif est de proposer une méthode combinant DE et HS de manière efficace et qui assure un bonne diversification. L'algorithme i-HSDE est donné dans l'algorithme 4.

Algorithm 4 : The iHDS algorithm for the MI-FAP

Require: an MI-FAP instance.
Ensure: An improved solution $Best$
1: Initialization : Create the population P with $PopSize$ solutions.
2: Evaluates the quality of every solutions within P using equation 3.1.
3: g $\leftarrow$ 0 ; {Counter of generation}
4: **while** (Stopping criterion is not met) **do**
5: **if** $(g\%2 == 0)$ **then**
6: Apply HS ;
7: Apply Mutation ;
8: **else**
9: Apply DE ;
10: **end if**
11: **if** Better solution is found **then**
12: Cpt $\leftarrow$ 0 ; {Cpt an integer that counts the number of successive no improvement of the best solution.}
13: **else**
14: Cpt $\leftarrow$ Cpt+1 ;
15: **if** $Cpt > NoImprovment$ **then**
16: Restart the population P ;
17: **end if**
18: **end if**
19: g $\leftarrow$ g+1 ;
20: **end while**

Nous présentons dans ce qui suit le principe de fonctionnement de l'algorithme proposé. La collaboration se fait en activant de manière alternée ces deux méthodes. Les deux méthodes partagent la même population P. A chaque génération la population est alors mise à jour par HS ou bien par DE. L'algorithme iHSDE que nous avons élaboré commence par activer l'algorithme HS. Ce dernier met à jour la population en appliquant les 5 étapes principales de HS présentées en section 5.2. Ensuite l'algorithme tente d'améliorer les performances à travers un opérateur de mutation. Pour réaliser cela nous proposons un opérateur simple et basique qui pour chaque variable de décisions compensant les solutions, décidera avec une probabilité $MP<1$ de perturber la valeur actuelle ou non. On obtient alors une population mise à jour P'. L'algorithme suivi par l'opérateur de mutation est donné dans l'algorithme 5. La population P' est ensuite transférée à l'algorithme DE qui à son tour mettra à jour la population dans le but de trouver une meilleure solution. Nous définissons dans la sous-section 5.5.1 les étapes

principales suivies par DE.
Lorsque DE se termine, l'algorithme transfère la population à HS. Si le processus n'enregistre pas d'amélioration après $NoImprovment$ de fois l'algorithme va alors réinitialiser les solutions et reprendre avec une nouvelle population P.

Algorithm 5 : The mutation operator within iHDS algorithm for the MI-FAP.

Require: P ; MP ;
1: **for** (i←0 ; $i < PopSize$; i←i+1) **do**
2: **for** (each decision variable composing the i^{th} solution $\in P$) **do**
3: r←random number between 0 and 1 ;
4: **if** $(r < MP)$ **then**
5: Change the frequency assigned to the TRX with another one ;
6: **end if**
7: **end for**
8: Evaluates the quality of this solution ;
9: **end for**

5.5.1. L'algorithme de l'évolution différentielle

L'évolution différentielle (DE) est une métaheuristique basée population apparue en 90. Proposée par Storn et Price pour résoudre l'ajustement par polynômes de Tchebychev (Tchebychev polynomial fitting problem) [63]. La stratégie suivie par DE pour résoudre un problème d'optimisation consiste à utiliser l'information contenue dans la population. La méthode manipule une population P de NP solutions générées aléatoirement. Différentes variantes de DE ont été suggérées par Price et al. [63] et sont classiquement appelées DE/x/y/z, où DE désigne l'évolution différentielle, x fait référence au mode de sélection de l'individu de référence ou l'individu cible (rand ou best) pour la mutation, y est le nombre de différenciations utilisées pour la perturbation du vecteur cible x et z désigne le schéma de croisement, qui peut être binomial ou exponentiel.
Le processus de DE se décline en trois opérations principales :

- **La mutation** : pour réaliser cette étape le processus crée pour chaque individu (targets individuals) de la $g^{ème}$ itération son individu mutant. Pour obtenir l'individu mutant, l'algorithme sélectionnera de manière aléatoire deux individus de la population P. Ensuite, la différence entre les variables de décision de ces deux solutions est calculée. Cette différence est pondérée et ajoutée à l'individu cible. Enfin pour obtenir l'individu mutant il existe plusieurs stratégies :
  - Rand/1 :

$$V_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}) \tag{5.4}$$

- Best/1 :

$$V_{i,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G}) \tag{5.5}$$

- Current to best/1 :

$$V_{i,G+1} = x_{i,G} + F.(x_{r1,G} - xr2, G) + F.(x_{best,G} - x_{i,G}) \tag{5.6}$$

- Best/2 :

$$V_{i,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{r3,G} - x_{r4,G}) \tag{5.7}$$

- Rand/2 :

$$V_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - xr3, G) + F.(x_{r4,G} - x_{r5,G}) \tag{5.8}$$

Les indices r1, r2, r3, r4 et r5 sont des entiers aléatoires tous différents. G correspond à la $G^{ème}$ itération. $F \in [0, 2]$ est une valeur constante, appelée differential weight, qui contrôle l'amplification de la variation différentielle de $X_{r1,g} - X_{r2,g}$.
La stratégie de mutation pour laquelle nous avons opté est la première indiquée ci dessus :

$$V_i = X_{i,G} + F * |X_{r1,G} - X_{r2,G}| \tag{5.9}$$

Tel que V_i correspond à l'individu obtenu après mutation, $X_{i,G}$ correspond à l'individu appelé target individu de la population P. $X_{r1,G}$ et $X_{r2,G}$ sont deux solutions distinctes sélectionnées aléatoirement de P pour produire l'individu mutant.

- **Le croisement** : Après la mutation un croisement est appliqué. Pour réaliser cette action, le processus combine l'individu target $X_{i,G}$ et son correspondant mutant $V_{i,G}$. L'individu issu de ce croisement est nommé individu d'essai $U_{i,G}$. Pour obtenir cet individu le processus choisira avec une probabilité CR $(1 - CR)$ pour chaque variable de décision j celle de l'individu target ou celle de l'individu mutant. Le croisement dans DE a pour but d'augmenter la diversité des vecteurs perturbés. Le nouveau vecteur $U_{i,G}$ est donné par la formule suivante 5.10 :

$$U_{i,j,G} = \begin{cases} V_{i,j,G} & \text{if rand(0,1)<CR} \\ X_{i,j,G} & \text{otherwise} \end{cases} \tag{5.10}$$

- **La sélection** : La sélection intervient juste après, pour choisir l'individu père X_i et son descendant U_i, le meilleur étant conservé pour la génération suivante.

5.6. Expérimentations et résultats numériques

Pour montrer l'efficacité des méthodes développées, nous avons mené notre étude sur des données réalistes de différentes tailles prises à partir d'un benchmark public. Les meilleurs résultats connus pour ces instances ont été obtenus par une recherche de tabous améliorée proposée par Montemanni et al. [58]. Les caractéristiques des instances sélectionnées sont dans le tableau 3.1 du chapitre précédent. En raison de la nature non déterministe des différents algorithmes, 20 essais ont été pris en compte pour chaque algorithme sur chaque instance. Pour rappel, le critère d'arrêt utilisé pour tous nos algorithme c'est le temps CPU. Ce dernier est fixé à 2400 secondes. Ce temps est le temps utilisé par [58] pour obtenir leurs bons résultats.

5.6.1. HS Vs. MHS Vs. Opp-HS

Nous présentons en première partie les résultats obtenus par HS, MHS et Opp-HS. Nous verrons par la suite les résultats obtenus par l'approche collaborative de HS et DE.
Les résultats des expérimentations sont présentés dans les tableau 5.1 à 5.6. Les différents tableaux présentent les statistiques descriptives des expérimentations réalisées. Ainsi la colonne Average, Best et worst indique la moyenne, la meilleure et la moins bonne solution trouvée par chaque algorithme sur 20 exécutions. Les meilleurs résultats, en termes de Average et Best, sont mis en évidence en gras. La dernière colonne indique le meilleur résultat connu. La première colonne (Instances) indique le nom de l'instance et la seconde (F) indique le nombre de fréquences possibles. Donnons quelques détails sur les paramètres des méthodes. Les valeurs des deux paramètres principaux de HS sont fixées expérimentalement : $HMCR$ = 0,9 et PAR =0,1 les paramètres utilisés sont ceux proposées dans configuration de base proposée par son auteur [34]. Nous avons fixé $HMS = 25$ pour les trois méthodes, $n_{Nouveau}$ = 20 pour (MHS) et (Opp-HS). Le nombre d'itérations pour la recherche locale utilisée dans (Opp-HS) est fixé à V, ce qui correspond au nombre de ($TRXs$) pour chaque instance.

TABLE 5.1. – Results obtained by the HS, MHS and Opp-HS algorithms with the AC-x-y instances datasets

Instances	|F|	Results	HS	(MHS)	(Opp-HS)	Metaheuristics best results
AC-45-17	7	Best	37	35	32	32
		Average	38.6	35.5	**32**	
		Worst	40	39	32	
		Std.	1.51	0	0	
AC-45-17	9	Best	18	17	15	15
		Average	19.2	18.35	**15**	
		Worst	20	19	15	
		Std.	0.83	0	0	
AC-45-25	11	Best	36	37	33	33
		Average	37.4	38	**33**	
		Worst	38	39	33	
		Std.	1.26	0	0	
AC-95-9	6	Best	45	45	35	31
		Average	49.2	46	**36**	
		Worst	50	47	37	
		Std.	1.41	0.67	0	
AC-95-17	21	Best	40	27	17	10
		Average	43.25	27.1	**17.63**	
		Worst	45	28	19	
		Std.	1.86	0.30	0.8	

TABLE 5.2. – Results obtained by the HS, MHS and Opp-Hs algorithms with the GSM-x instances datasets

Instances	|F|	Results	HS	(MHS)	(Opp-HS)	Metaheuristics best results
GSM-93	9	Best	47	50	41	32
		Average	61.5	51.75	**43.5**	
		Worst	65	53	44	
		Std.	2.52	1.03	0.89	
GSM-93	13	Best	18	12	11	7
		Average	18.8	13.8	**11.8**	
		Worst	20	16	15	
		Std.	1.31	0.63	0.51	
GSM-246	21	Best	305	309	130	79
		Average	312.66	312.66	**132**	
		Worst	315	314	135	
		Std.	1.03	2.32	1.71	

TABLE 5.3. – Results obtained by the HS, MHS and Opp-HS algorithms with the Test instances datasets

Instances	|F|	Results	HS	(MHS)	(Opp-HS)	Metaheuristics best results
Test282	61	Best	98	95	84	51
		Average	108	100.63	**86.85**	
		Worst	110	111	100	
		Std.	3.6	5.18	2.26	
Test95	36	Best	15	13	11	8
		Average	16.61	14	**11.5**	
		Worst	19	15	12	
		Std.	1.26	0.00	0.00	

TABLE 5.4. – Results obtained by the HS, MHS and Opp-HS algorithms with the instances datasets

Instances	|F|	Results	HS	(MHS)	(Opp-HS)	Metaheuristics best results
P06-5	11	Best	245	240	140	133
		Average	251.87	245.23	**147.26**	
		Worst	256	248	148	
		Std.	3.81	2.61	0.45	
P06-3	31	Best	236	230	130	115
		Average	249.5	238.6	**135.1**	
		Worst	258	249	145	
		Std.	9.83	0.00	0.31	
P06b-5	31	Best	39	36	29	25
		Average	40.15	36.2	**29.1**	
		Worst	43	37	30	
		Std.	6.56	0.41	0.31	
P06b-3	31	Best	123	126	122	112
		Average	131.5	131	**127**	
		Worst	133	140	130	
		Std.	4.95	4.64	3.18	

TABLE 5.5. – Results obtained by the HS, MHS and Opp-HS algorithms with the instances GSM2 datasets

Instances	|F|	Results	HS	(MHS)	(Opp-HS)	Metaheuristics best results
GSM2-184	39	Best	6994	6534	5998	5250
		Average	7475.4	6554.34	**6180.71**	
		Worst	7835	6886	6826	
		Std.	306.73	281.50	191.47	
GSM2-227	29	Best	82586	78586	77586	57731
		Average	87626.91	79840.82	**78721**	
		Worst	89531	85423	70566	
		Std.	2264.35	3668.68	1096.28	
GSM2-272	34	Best	88664	75568	75150	53080
		Average	91760.9	74409.9	**77888.3**	
		Worst	97806	76405	79755	
		Std.	3164.63	4466.47	1477.70	

TABLE 5.6. – Results obtained by the HS, MHS and Opp-HS algorithms with the instances r- datasets

Instances	|F|	Results	HS	(MHS)	(Opp-HS)	Metaheuristics best results
1-1-50-75-30-2-50	5	Best	1317	1298	1402	1242
		Average	1378.33	**1327.25**	1451	
		Worst	1453	1389	1485	
		Std.	44.12	38.33	9.12	
1-2-50-75-30-4-50	9	Best	703	672	670	665
		Average	753.61	697.69	**674.18**	
		Worst	781	766	678	
		Std.	27.78	28.77	2.89	
1-3-50-75-30-0-50	7	Best	320	295	206	194
		Average	340.61	317.69	**206.76**	
		Worst	349	332	208	
		Std.	8.64	14.40	1.09	
1-4-50-75-30-2-1	6	Best	88	76	74	70
		Average	90	78	**76.2**	
		Worst	91	79	78	
		Std.	0.89	0.89	2.25	

- **AC-xy** : le tableau 5.1 montre que pour les instances de petite taille, les variantes proposées pour améliorer HS donnent de meilleurs résultats que ceux obtenus par HS. On observe aisément que la méthode Opp-HS surpasse les deux méthodes basées sur HS. On constate également que contrairement à la méthode HS et MHS, Opp-HS parvient à atteindre les résultats de l'état de l'art pour les trois premières instances. Il est à noter que les modifications apportées à HS à travers Opp-HS ont permis d'obtenir le meilleur résultat connu au cours des 20 tests effectués et d'enregistrer ainsi un Sdt parfait égal à 0 pour les trois premiers jeux de données, ce qui indique une cohérence et une capacité à traiter des instances petites tailles. Pour les autres instances aucune des méthodes n'atteint les résultats atteints par les travaux de références.
- **GSM-x instances** et **Instance Test 95 et Test 282** : Les résultats des tests effectués sur les instances GSM-x sont présentés dans le tableau 5.2 et le tableau 4.4 présente les résultats des trois méthodes obtenus sur les instances Test. Les résultats obtenus pour les instances de GSM de même que les instances Test indiquent que, les méthodes MHS et Opp-HS sont plus performantes que HS. Les meilleurs résultats sont produits par la méthode Opp-HS. Les valeurs de Sdt sont relativement petits. On peut observer que les variantes de HS proposées génèrent de meilleurs résultats que la méthode HS. Les résultats de Opp-HS se rapprochent de ceux de l'état de l'art pour l'instance GSM-93(13) ainsi que pour l'instance Test95.
- **Instances P06-z et P06b-z** : On observe selon les résultats présentés dans le tableau 5.4 que la méthode Opp-HS a donné de meilleurs résultats que les deux autres méthodes. En moyenne, les méthodes Opp-HS et MHS ont toutes deux de meilleures résultats que HS. Il a été constaté néanmoins

que dans tous les cas nos méthodes ne parviennent pas à réaliser les performances relatées par la littérature.

- **Les instances GSM2-x** : Les résultats présentés dans le tableau 5.5 indiquent que Opp-HS a produit de meilleurs résultats, pour les différentes instances, par rapport aux deux autres méthodes. On peut également noter que l'écart entre les résultats de Opp-HS et ceux de MHS est de moindre importance que celui observé entre Opp-HS/HS et MHS/HS. Le $std.$ enregistré pour les différentes instances GSM2 est relativement petit. Comme on peut le noter, les résultats obtenus par les méthodes proposées sur les instances GSM2 montrent que toutes les méthodes n'ont pas abouti aux meilleurs résultats connus et ont convergé vers un mauvais optimum local. En effet, ces instances sont difficiles et les divers éléments qui ont été incorporés dans le schéma de base de l'algorithme HS n'ont pas été aussi efficace face à des instances de grandes tailles ou le nombre d'optima locaux est assez important.
- **L'instance r1-r2-sxw-ms-mp** : les résultats présentés dans le tableau 5.6 montrent que, pour l'instance 1-1-50-75-30-2-50, la MHS a produit de meilleurs résultats que les autres méthodes. Toutefois, les résultats obtenus par cette méthode sont assez proches de ceux de la HS. Comme on peut le constater Opp-HS n'a pas réussit a donné de bons résultats pour cette instances. Les résultats obtenus restent tout de même non satisfaisants par rapport à l'état de l'art. On observe également que tout comme pour le reste des instances du benchmark Opp-HS donne les meilleurs résultats. Les deux méthodes améliorées sont plus performantes que HS dans la plus part des cas. Pour l'instance 1-4-50-75-30-2-1, Opp-HS on peut remarquer que la méthode parvient à des résultats qui sont très proches des meilleurs résultats connus.

En résumé, on constate aisément que les résultats obtenus par Opp-HS sont supérieurs à ceux obtenus par HS MHS dans tous les cas, sauf dans le cas 1-1-50-75-30-2-50 où l'algorithme se perd dans un optimum local et ne peut pas s'échapper. La méthode Opp-HS parvient à atteindre meilleurs résultats connus pour les instances AC-45-17 (7), AC-45-17 (9), AC-45-25 et 1-4-50-75-30-2-1. En outre, Opp-HS a trouvé des résultats proches de l'état de l'art pour l'instance (AC-95-9) mais a échoué dans les autres cas. La méthode MHS se montre plus efficace sur toutes les instances en comparaison à HS.
Bien que nos méthodes n'aient pas réussi à trouver les résultats obtenus dans [58] pour toutes les instances , les résultats confirment que l'introduction de la recherche locale et l'apprentissage basé sur l'opposé dans Opp-HS améliorent les performances de la méthode étudiée et réduisent l'écart avec les meilleurs résultats trouvés dans le état de l'art pour certaines instances, tout en permettant de l'atteindre dans certains cas. Cela prouve que l'introduction de la recherche locale et de l'OPBL (dans la méthode Opp-HS a un impact positif sur l'amélioration

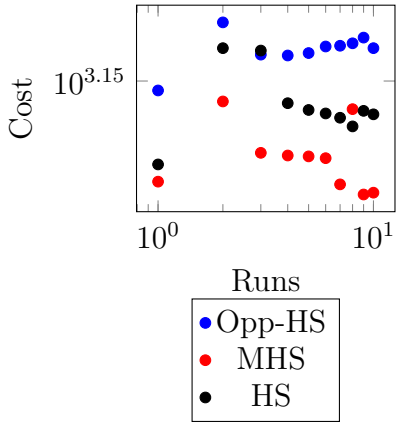


FIGURE 5.2. – Details of experiments on the instance 1-1-50-75-30-2-50

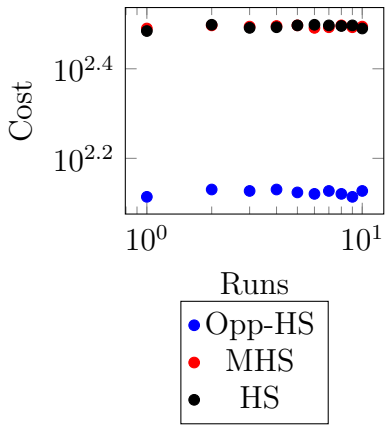


FIGURE 5.3. – Details of experiments on the instance GSM-246

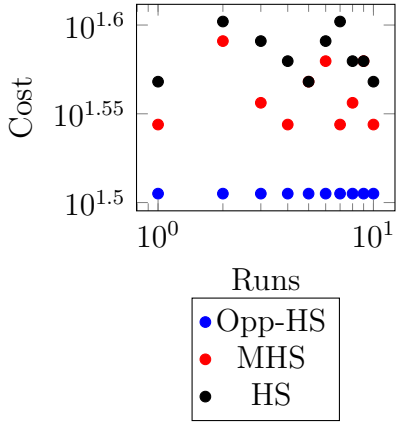


FIGURE 5.4. – Details of experiments on the instance AC45-17(7)

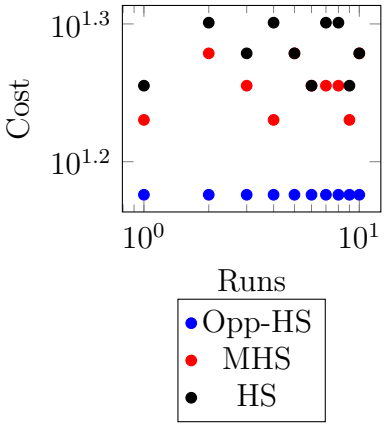


FIGURE 5.5. – Details of experiments on the instance AC45-17(9)

des résultats. On peut dire que Opp-HS engendre un bon équilibre entre les deux concepts principaux régissant les performances de toute métaheuristique à savoir l'intensification et la diversification. Ceci est possible par l'ajout de LS qui est appelé lorsque HS échoue à améliorer les résultats, permettant ainsi d'intensifier la recherche autour de chaque solution. Cela garantit une meilleure exploitation. En outre, la présence de l'OPBL a permis l'exploration de divers points dans l'espace de recherche ce qui apporte la diversité et évite de rester bloqué dans un optimum local.
Les résultats et les interprétations présentés ci-dessous permettent de conclure que la méthode Opp-HS est intéressante et pourrait être beaucoup plus efficace avec une meilleure mise en œuvre et une étude plus approfondie de ses paramètres.

5.6.2. Analyse statistique

Nous avons voulu vérifier si les différences entre les solutions trouvées par les trois méthodes étaient significatives. A cet effet nous effectuons, comme pour la recherche VNS, le test non paramétrique de Friedman pour déterminer si les algorithmes sont statistiquement différents.
Nous avons effectué le test de Friedman sur l'ensemble des données testées en vue de détecter toutes différences entre les trois algorithmes Harmony search proposés. Nous avons effectué le test avec le R logiciel sous le niveau de signification alpha = 0,05.
Les résultats du test sont les suivants : Friedman chi-squared = 32.12, df = 2, p-value = 1.06e-07. Comme on peut le constater, le test montre une différence significative (valeur p <0,05), de sorte que l'hypothèse nulle peut être rejetée car la valeur p est inférieure à 0,05. Les résultats sont donc différents. L'hypothèse nulle du test de Friedman étant rejetée, il est nécessaire de déterminer quels algorithmes diffèrent les uns des autres. À cette fin, nous avons effectué un test post-hoc de Nemenyi afin de comparer différentes paires d'algorithmes. Les résultats du test post-hoc de Nemenyi sont résumés dans Tableau 7.7. Au vu de ces résultats, nous pouvons conclure que les algorithmes comparés sont statistiquement différents. En effet, on peut voir que le Opp-HS diffère considérablement (p <0,05) du HS et de MHS et que ce dernier diffère de manière significative du HS (p <0,05). Il est donc évident que les modifications apportées à l'algorithme HS, par le biais de ces deux variantes, ont eu un impact réel sur les performances de l'algorithme.

TABLE 5.7. – Nemenyi post hoc tests

Méthodes	HS	MHS
MHS	0.0094	–
Opp-HS	5.3e-08	0.0190

5.6.3. HS Vs. DE Vs i-HSDE

Nous consacrons la deuxième partie de cette section à la présentation et à l'analyse des résultats obtenus suite aux expérimentations effectuées pour les méthodes HS, DE et i-HSDE. Nous comparons les différentes méthodes entre elles ainsi qu'aux meilleurs résultats trouvés dans la littérature. Les tableaux de 5.8 à 5.13 résument les différents résultats sur les différentes instances du MI-FAP.

Pour les paramètres de DE, nous avons fixé la taille de la population à $4 * d$ individus tel que d le nombre de TRX du réseau. Les paramètres $F = 0.5$ et $CR = 0.9$, des valeurs généralement recommandées.

Nous pouvons observé des résultats numériques présentés dans les tableaux de 5.8 à 5.13 que la méthode iHDS produit des résultats de bien meilleure qualité par rapport à ceux de DE et HS pour toutes les instances. Nous observons également que la moins bonne valeur enregistrée pour le iHDS est toujours de meilleure qualité que celle enregistrée par les méthodes HS et DE. Cela signifie que cette méthode collaborative permet au processus de recherche d'échapper aux mauvaises régions pour aboutir à de meilleurs résultats.

On peut observer aussi que les résultats obtenus par la méthode HS sont nettement meilleurs que ceux donnés par la méthode DE dans de nombreux cas. Par contre, pour les instances le GSM-93 et AC-95-9, on peut voir que DE produit des résultats assez proches en terme de moyenne voir meilleur, et similaire en terme de Best. Pour l'instance AC-45-17 (9), les résultats montrent que la méthode HS donne de meilleurs résultats en termes de average comparables pour le Best. L'écart-type est relativement petit pour les trois méthodes avec toutes les instances. Cette petite valeur indique que les méthodes implémentées présentent une stabilité lors du traitement de ces instances. Même si la méthode HS et DE n'a pas réussi à obtenir d'aussi bons résultats que ceux obtenus par l'algorithme iHDS, en convergeant vers un mauvais optimum local, la petite valeur de l'écart type enregistré indique que les résultats DE et HS ne sont généralement pas dispersés et relativement centré dans la même région.

En comparant nos résultats avec l'état de l'art, nous pouvons relever que les deux méthodes DE et HS que nous avons mises en œuvre n'ont pas réussi à trouver d'aussi bons résultats que ceux de l'état de l'art. En effet les méthodes convergent et stagnent dans un optimum local assez mauvais. Néanmoins, on peut constater que l'iHDS que nous proposons réussit à améliorer ces résultats, atteignant ainsi de meilleurs résultats et se rapprochant de ceux de l'état de l'art pour les différentes instances. Nous remarquons aussi que, dans ces deux cas, le processus aboutit dans certains cas à des résultats comparables.

TABLE 5.8. – Results obtained by the HS, DE and iHSDE algorithms with the AC-x-y instances datasets

Instances	|F|	Results	HS	DE	iHSDE	Metaheuristics best results
AC-45-17	7	Best	37	40	<u>34</u>	32
		Average	38.6	42.4	**34.9**	
		Worst	40	47	37	
		Std.	1.51	2.22	0.87	
AC-45-17	9	Best	18	17	<u>16</u>	15
		Average	19.2	24.6	**16.9**	
		Worst	20	21	18	
		Std.	0.83	2.01	0.56	
AC-45-25	11	Best	36	35	<u>33</u>	33
		Average	37.4	39.8	**33.2**	
		Worst	39	45	35	
		Std.	1.14	3.32	1.13	
AC-95-9	6	Best	40	39	<u>31</u>	31
		Average	46.1	43	**31.7**	
		Worst	58	46	32	
		Std.	5.84	3.26	0.48	
AC-95-17	21	Best	40	39	<u>26</u>	10
		Average	43.25	42.1	**26.63**	
		Worst	45	43	28	
		Std.	1.86	0.30	0.8	

Nos résultats démontrent que l'approche collaborative proposée est intéressante pour traiter le MI-FAP. En effet la stratégie adopté a réussi à améliorer de loin les résultats obtenus par HS avec toutes les instances réduisant ainsi l'écart avec les résultats de l'état de l'art. iHDS atteint les meilleurs résultats connus pour deux instances données et il est très proche pour les autres instances. Néanmoins, la méthode n'a pas abouti à des résultats intéressants pour certaines instances telles que les instances GSM-x et GSM2 pour lesquelles nous enregistrons un mauvais optimum local.

Même si notre iHDS n'atteint pas les résultats de l'état de l'art, les résultats obtenus confirment que la stratégie de collaboration entre le HS et le DE de manière alternative, ainsi que le renforcement par un opérateur de mutation et le redémarrage adaptatif agit sur la diversité et booste les performances des méthodes étudiées et diminue l'écart avec les résultats publiés précédemment. Cela prouve que cette collaboration agit positivement en stimulant le processus de recherche.

En effet, l'utilisation des méthodes HS et DE permet une exploitation plus large de l'information au sein de la population, de sorte que le processus intensifie la recherche autour des solutions menant à de meilleures performances.

TABLE 5.9. – Results obtained by the HS, DE and iHDSE algorithms with the GSM-x instances datasets

Instances	|F|	Results	HS	DE	iHSDE	Metaheuristics best results
GSM-93	9	Best	47	47	38.2	32
		Average	61.5	50	**37**	
		Worst	94	57	40	
		Std.	21.82	3.05	1.13	
GSM-93	13	Best	18	19	15	7
		Average	18.8	20.8	**16.8**	
		Worst	20	21	17	
		Std.	1.31	0.63	0.51	
GSM-246	21	Best	305	505	242	79
		Average	312.66	512.2	**248.3**	
		Worst	317	518	252	
		Std.	6.42	4.87	3.59	

TABLE 5.10. – Results obtained by the HS, DE and iHSDE algorithms with the Test instances datasets

Instances	|F|	Results	HS	DE	iHSDE)	Metaheuristics best results
Test282	61	Best	98	99	87	51
		Average	108	105.63	**89.85**	
		Worst	110	111	100	
		Std.	3.6	5.18	2.26	
Test95	36	Best	15	17	12	8
		Average	16.61	18	**12.5**	
		Worst	19	19	13	
		Std.	1.26	0.00	0.00	

TABLE 5.11. – Results obtained by the HS, DE and iHSDE algorithms with the instances datasets

Instances	|F|	Results	HS	DE	iHSDE	Metaheuristics best results
P06-5	11	Best	245	247	143	133
		Average	251.87	248.23	**150.26**	
		Worst	256	249	158	
		Std.	3.81	2.61	0.45	
P06-3	31	Best	236	529	172	115
		Average	249.5	539.4	**177.4**	
		Worst	258	546	183	
		Std.	9.83	5.60	2.98	
P06b-5	31	Best	39	46	35	25
		Average	40.15	46.2	**38.1**	
		Worst	43	47	39	
		Std.	6.56	0.41	0.31	
P06b-3	31	Best	123	138	122	112
		Average	131.5	140	**127**	
		Worst	133	145	130	
		Std.	4.95	4.64	3.18	

TABLE 5.12. – Results obtained by the HS, DE and iHSDE algorithms with the instances GSM2 datasets

Instances	|F|	Results	HS	DE	iHSDE	Metaheuristics best results
GSM2-184	39	Best	6994	7533	6988	5250
		Average	7475.4	7554.34	**7170.71**	
		Worst	7835	7886	7206	
		Std.	306.73	281.50	191.47	
GSM2-227	29	Best	82586	88589	81586	57731
		Average	87626.91	89840.82	**818721**	
		Worst	89531	89843	82566	
		Std.	2264.35	3668.68	1096.28	
GSM2-272	34	Best	88664	76568	65150	53080
		Average	91760.9	76409.9	**67888.3**	
		Worst	97806	77405	69755	
		Std.	3164.63	4466.47	1477.70	

TABLE 5.13. – Results obtained by the HS, DE and iHSDE algorithms with the instances r- datasets

Instances	|F|	Results	HS	(DE)	(iHSDE)	Metaheuristics best results
1-1-50-75-30-2-50	5	Best	1317	1326	1251	1242
		Average	1378.33	1396.5	**1288**	
		Wort	1453	1506	1328	
		Std.	44.12	48.51	28.05	
1-2-50-75-30-4-50	9	Best	703	772	680	665
		Average	753.61	797.69	**684.18**	
		Worst	781	866	688	
		Std.	27.78	28.77	2.89	
1-3-50-75-30-0-50	7	Best	320	385	206	194
		Average	340.61	387.69	**210.76**	
		Worst	349	399	212	
		Std.	8.64	14.40	1.09	
1-4-50-75-30-2-1	6	Best	78	80	72	70
		Average	81	87.5	**75.54**	
		Wort	80	95	79	
		Std.	0.89	4.32	2.08	

L'ajout de l'opérateur de mutation et la présence d'un redémarrage adaptatif basé sur OPBL permettent une grande exploration de l'espace de recherche et apportent suffisamment de diversité, ce qui est important pour traiter un problème difficile tel que le MI-FAP pour éviter l'attraction des optima locaux. Ainsi, l'iHDS établit un équilibre entre l'intensification et la diversification qui constitue l'épine dorsale de toute métaheuristique.
Les premières observations données ci-dessus mènent à la conclusion préliminaire que la méthode iHDS montre une supériorité par rapport à DE et HS. C'est donc une approche intéressante qui peut être explorée et améliorée.

5.6.4. Analyse statistique

Nous avons voulu vérifier si les différences entre les solutions trouvées par les trois méthodes étaient significatives. A cet effet nous effectuons le test non paramétrique de Friedman pour déterminer si les algorithmes sont statistiquement différents. Nous avons effectué le test de Friedman sur les jeux de données afin de détecter toutes différences entre les trois algorithmes proposés. Nous avons effectué le test avec le R logiciel sous le niveau de signification alpha = 0,05.
Les résultats du test sont les suivants : Friedman chi-squared = 28.727, df = 2, p-value = 5.78e-07. Comme on peut l'observer, le test montre la signification (valeur p <0,05), de sorte que l'hypothèse nulle peut être rejetée car la valeur p est inférieure à 0,05. Les résultats sont donc différents. L'hypothèse nulle du test de Friedman étant rejetée, il est nécessaire de déterminer quels algorithmes diffèrent les uns des autres. À cette fin, nous avons effectué un test post-hoc de Nemenyi afin de comparer différentes paires d'algorithmes. Les résultats du test post-hoc de Nemenyi sont résumés dans Tableau 5.14. Au vu de ces résultats, nous pouvons conclure que les algorithmes comparés sont statistiquement différents. En effet, on peut voir que le i-HSDE diffère considérablement (p <0,05) de HS et de DE. Néanmoins on constate que ce dernier ne diffère pas de HS (p >0,05). Il est donc évident que la collaboration entre les deux méthodes HS et DE à savoir les DE a eu un impact réel sur les performances.

TABLE 5.14. – Nemenyi post hoc tests

Méthodes	HS	DE
DE	0.44955	–
i-HSDE	0.00026	8.9e-07

5.7. Synthèse

Le présent travail avait pour objet de résoudre efficacement le MI-FAP en adaptant harmony search. Nous avons privilégié cette méthode pour sa relative récence. Nous avions également comme objectif l'amélioration des performances de cette méthode face au MI-FAP à travers des stratégie différentes qui sont la mémetic, l'adaptation et la collaboration ainsi que le concept de OPBL. Nous avons donc proposé trois améliorations de HS destinées à améliorer ses performances. Nos expériences ont clairement montré que les améliorations expérimentées donnent de meilleurs résultats que la méthode HS de base. Les résultats obtenus sont encourageants. Néanmoins, certaines limites subsistent. Donc, d'autres travaux sont encore nécessaires pour régler et éliminer ces limitations. Nous avons comme perspectives en premier lieu d'étudier la possibilité de combiner de manière plus manière efficace le DE et la méthode HS. D'autres travaux

seront également dédiés sur une combinaison de LS, d'OPBL et de HS plus efficace, tout en étudiant la possibilité d'ajouter d'autres combinaisons dans le but d'améliorer l'efficacité de l'algorithme. Il est impératif que nos travaux futurs soient orientés de manière à parfaire les méthodes et à régler les inconvénients qui empêchent les algorithmes basés sur la recherche harmonique d'atteindre les résultats affichés dans la littérature pour toutes les instances.

6. L'algorithme de Jaya pour le MI-FAP

Sommaire

6.1. Introduction

Ce chapitre est dédié à la présentation de l'étude consacrée à l'algorithme de Jaya (JA). En outre ce chapitre présente les tentatives d'amélioration des performances de l'algorithme de Jaya pour résoudre le MI-FAP. Pour réaliser notre étude nous avons tout d'abord commencer par adapter l'algorithme de Jaya au MI-FAP. Nous proposons ensuite une version memetic de Jaya (MJ). Ensuite nous proposons d'améliorer les performances de Jaya à travers un algorithme adaptatif, nous développons alors Adaptive Jaya Algorithm (AJA). Pour ce faire, nous détaillons dans ce que suit les différents variantes proposées et nous présenterons également les expérimentations effectuées pour valider nos algorithmes. Nous donnerons enfin une conclusion sur le travail dédié à Jaya.

6.2. L'algorithme de Jaya pour le MI-FAP

Avant de voir en détail l'algorithme de Jaya et ses spécificités, il est utile de connaître le sens du mot Jaya. Ce dernier est un mot sanskrit qui signifie victoire. Ce mot a été utilisé pour désigner l'algorithme de Jaya qui est une méthode d'optimisation récente développée par Rao[66]. C'est une métaheuristique basée population. Rao a démontré les performances de sa méthode en appliquant Jaya sur des problèmes d'optimisation avec et sans contraintes (21 problèmes avec

contraintes) et quatre problèmes d'optimisation liés à la mécanique [66]. L'élément le plus intéressant dans l'algorithme de Jaya réside dans le fait qu'il n'a pas de paramètre à régler. Cette méthode est simple à implémenter et demande seulement à fixer la taille de la population.
L'algorithme de Jaya est proche de l'évolution différentielle. En effet, sa méthode se base également sur l'information contenue dans la population. Plus particulièrement Jaya utilise les distances entre les variables de décisions. Plus en détail, Jaya utilise l'information contenue dans la meilleure et la moins bonne solution pour tenter d'obtenir une éventuelle meilleure solution.
Ainsi, le processus va créer de nouvelles solutions en se basant sur la formule 6.1. Cette dernière aura donc à calculer la valeur $f_j^{'}$ de la j^{th} variable de décision de la solution $X^{'}$ en utilisant la différence entre les variables best, worst et une solution j de la population afin de créer de nouvelles solutions :

$$f_{j,k',i}^{'} = f_{j,k,i} + r_{1,j,i} * |f_{j,best,i} - f_{j,k,i}| - r_{2,j,i} * |f_{j,worst,i} - f_{j,k,i}| \tag{6.1}$$

Tel que :

- $f_{j,k,i}$ C'est la valeur de la j^{th} variable de décision de la k^{th} solution à la i^{th} itération
- $f_{j,best,i}$ correspond à la j^{th} variable de décision de la best solution à la i^{th} itération.
- $f_{j,worst,i}$ correspond à la j^{th} variable de décision de la worst solution à la i^{th} itération.
- $r_{1,j,i}$, $r_{2,j,i}$ nombre aléatoire pris dans l'intervalle $[0, 1]$ pour la j^{th} variable de décision à la i^{th} itération.

Dans l'équation 6.1 les termes $r_{1,j,i}$ *|$f_{j,best,i}$-$f_{j,k,i}$| indiquent la tendance de la solution de s'approcher davantage de la meilleure solution. Quant au terme $r_{2,j,i}$*|$f_{j,worst,i}$-$f_{j,k,i}$| il indique que la solution se dirigera davantage vers la moins bonne solution (worst solution.)
Les étapes de l'algorithme Jaya sont illustrées dans l'algorithme 6.

Algorithm 6 : The Jaya algorithm for the MI-FAP

Require: an MI-FAP instance.
Ensure: An improved solution $Best$
1: Initialization of the population P with $PopSize$ solutions
2: **while** (Stopping criterion is not met) **do**
3: Identify the best and the worst solutions from P.
{Create $PopSize$ new Solutions}
4: **for** ($i = 0$; i<$PopSize$; ++i) **do**
5: **for** (each decision variable within $X_i^{'}$) **do**
6: calculate its value using equation 6.1.
7: **end for**
8: Calculate Cost($X^{'}$) using equation 3.1.
9: **if** ($Cost(X^{'})$<$Cost(X)$) **then**
10: X=$X^{'}$;
11: **end if**
12: **end for**
13: **end while**

6.3. Memetic-Jaya pour le MI-FAP

Dans le but d'améliorer les performances de l'algorithme Jaya face au problème MI-FAP nous proposons d'explorer les possibilités d'une méthode mémétique. Nous proposons alors l'algorithme Memetic Jaya Algorithm (MJA). En effet les algorithmes mémétiques ont prouvé leur efficacité à résoudre des problèmes complexes d'optimisation et sont des outils puissants pour améliorer les résultats d'un algorithme à population classique.
Nous proposons d'intégrer à Jaya une recherche locale (LS) qui permettra d'améliorer sa capacité à résoudre le MI-FAP. LS est alors appelé à chaque fois qu'une nouvelle solution $X^{'}$ est créée en vue d'améliorer éventuellement sa qualité. La méthode LS est ajoutée après la ligne 8 de l'algorithme 6.
La méthode LS que nous utilisons est une procédure itérative qui prend en entrée la solution créée$X^{'}$. Ensuite un voisin de cette solution est générée en modifiant aléatoirement une des fréquences affectée à un TRX. Le processus valide la solution si elle est de meilleure qualité. Le processus est ainsi répété à travers un certain nombre d'itérations.
L'algorithme MJA est présenté dans l'algorithme 7

Algorithm 7 : The Memetic Jaya algorithm for the MI-FAP

Require: an MI-FAP instance.
Ensure: An improved solution $Best$
1: Initialization of the population P with $PopSize$ solutions
2: **while** (Stopping criterion is not met) **do**
3: Obtain new solution using equation 6.1.
4: Calculate fitness of the new solution.
5: Improve the new solution via Local Search.
6: Replace with the new solution if it is better.
7: **end while**

6.4. Adaptive-Jaya pour le MI-FAP

Toujours avec pour objectif d'améliorer les résultats de Jaya face aux différentes instances du problème MI-FAP que nous traitons,nous proposons de concevoir une méthode basée sur Jaya avec une stratégie adaptative. Cette algorithme est nommé adaptive Jaya Algorithm (AJA). Ainsi, des améliorations ont été apportées à l'algorithme de base de Jaya pour permettre un meilleur équilibre entre les phases de diversification et d'intensification. Afin d'éviter à l'algorithme de stagner nous proposons alors une combinaison entre la force qu'offre le concept d'OPBL le tout greffer sur un schéma adaptatif.
L'algorithme AJA disposera de trois moyens pour être efficace :

1. Premièrement, l'utilisation d'une procedure recherche locale (LS) :

↓

   - Manipule une seule solution.
   - Utilisée dans le but d'accélrer la recherche.

2. Deuxièment, l'utilisation de l' Opposition based learning (OPBL) concepte :

↓

   - concept introduit en 2005 by Tizhoosh [69].
   - utilisation de ce concept pour intensifier, diversifié et assurer une large exploration de l'espace de recherche.
   - OPBL permet d'exploiter l'information contenue dans une solution et sa solution opposée.
   - OPBL augmentera les possibilités d'exploration de l'espace de recherche et éviter la convergence prématurée vers un optimum local. En effet pour trouver de meilleures solutions, la considération simultanée d'une direction et son opposé peut aider à résoudre un problème plus efficacement.

- L'OPBL a été utilisé pour résoudre des problèmes d'optimisation assez signifiants nous citons quelque travaux [53, 71, 64].
- Plus d'information sont disponibles concernant l'OPBL et son application aux problèmes optimisations ainsi qu'une bonne revue de la littérature est proposée par Xu et.al [74].

3. Finalement, l'application d'un mécanisme de ré-initialisation :

$$\downarrow$$

- Mécanisme qui permet de ré-initialiser la population.
- Utilisation d'une schéma adaptative pour pour éviter de stagner.
- Le processus redémarrera la population à chaque $NombreGen$ de fois toujours dans le but d'éviter la stagnation.

En fonction de l'état de la recherche, AJA dispose alors de ces trois options. Le processus démarre d'une population (P) initiale générée aléatoirement, AJA applique ensuite le schéma classique de Jaya pour mettre à jour la population jusqu'à la ligne 12 de l'algorithme 6). Ensuite la méthode bascule vers la partie adaptative pour aboutir à de meilleures solutions. Le processus doit alors vérifier si une meilleure solution a été obtenue et agir en conséquence :

- If pas de meilleure solution trouvée, Then appliquer la recherche locale sur chaque solution de P.
- If pas de meilleure de solution trouvée après $NoImprovment$ alors Then exploration des solutions opposées : construction de Opp_P tel que Opp_P contient toutes les solutions opposées à celles contenues dans P. Ainsi, chaque variable de décision est remplacée par sa valeur opposée. Pour rappel l'opposé d'une fréquence : f_{ij} d'une solution X est définie comme suit : considérons $f_{ij} \in [1, NF]$, l'opposée f'_{ij} de f_{ij} est donnée par : $f'_{ij} = 1 + NF - f_{ij}$. La procédure OPBL agit de la manière suivante :
  - Trouver l'opposé OX de chaque solution X dans P
  - Evaluer la qualité de OX
  - If $Cost(OX)$ est meilleure que $Cost(X)$ Then remplacer X=OX.
- If $NombreGen$ est atteint Then Ré-initialiser P
  - Inject les meilleures solutions dans P.

6.5. Expérimentations et résultats numériques

Pour bien évaluer nos méthodes, nous comparons les trois méthodes basées sur Jaya entre elles. Nous les comparons également au best known (meilleur résultat trouvé dans la littérature). Pour réaliser une comparaison équitable, nous utilisons comme critère d'arrêt le temps CPU. Nous le fixons à 2400 secondes. Cette valeur a été spécialement choisie car il s'agit du critère d'arrêt utilisé par les chercheurs ayant travailler sur ces instances. La valeur des paramètres

est fixé comme suit : $PopSize = 4 * d$(avec d le nombre de $TRXs$) pour les trois méthodes, le nombre d'itérations de la recherche locale ($IterLS = d * NF$), $NoImprovement = d$ et $NombreGen$: est spécifique à chaque instance et égal au nombre de contraintes.

D'après les résultats présentés dans les tableaux 6.1 à 6.6 nous observons que l'algorithme de Jaya avec son schéma de base ne donne pas de très bons résultats et cela avec toutes les instances. Les résultats indiquent que l'algorithme Jaya pur souffre de son manque de diversité, il est donc bloqué dans un mauvais optimum local.
A partir des résultats présentés dans les tableaux 6.1 à 6.6, nous observons que MJA et AJA donnent de meilleurs résultats que l'algorithme Jaya pur pour toutes les instances considérées. Nous voyons que la variante MJA réussit à améliorer les résultats par rapport à l'algorithme Jaya pur pour toutes les instances. Les résultats montrent que la méthode AJA est supérieure à la méthode Jaya pour toutes les instances. De plus, on peut constater que pour les instances (AC-45-17 à 7 fréquences et 95-5), AJA et MJA obtiennent les mêmes résultats. Les deux méthodes enregistrent Std égal à 0, ce qui est parfait. Pour l'instance AC-45-17 à 9 fréquences, nous constatons que l'AJA et la MJA obtiennent des résultats très proches en moyenne et enregistrent les mêmes Best résultats. Néanmoins, les résultats montrent que la MJA n'enregistre pas un écart type qui est proche de zéro, la méthode se perd dans un optimum local très proche dans une des exécutions 20 essais. Contrairement à la méthode AJA. Pour les exemples 1-4-50-75-30-2-1, les résultats montrent que la méthode AJA donne un résultat légèrement meilleur par rapport à la MJA. La méthode AJA enregistre un écart type parfait. L'écart-type de la MJA est relativement faible. Dans les autres cas, le résultat montre clairement que la méthode AJA est supérieure aux résultats MJA.

TABLE 6.1. – Results obtained by the JA, MJA and AJA algorithms with the AC-x-y instances datasets

Instances	|F|	Results	JA	(MJA)	(AJA)	Metaheuristics best results
AC-45-17	7	Best	51	**32**	**32**	32
		Average	59.5	**32**	**32**	
		Worst	66	32	32	
		Std.	4.41	0	0	
AC-45-17	9	Best	36	**15**	**15**	15
		Average	39	15.1	**15**	
		Worst	42	16	15	
		Std.	1.82	0.31	0	
AC-45-25 (45)	11	Best	67	36	**33**	33
		Average	72	40.6	**33**	
		Worst	75	43	33	
		Std.	36.98	20.87	0	
AC-95-9 (95)	6	Best	76	**31**	**31**	31
		Average	80.6	**31**	**31**	
		Worst	85	31	31	
		Std.	41.39	0	0	
AC95-17 (95)	15	Best	121	41	**39**	10
		Average	126.5	42.5	**41.4**	
		Worst	133	44	43	
		Std.	4.23	1.09	1.51	

TABLE 6.2. – Results obtained by the JA, MJA and AJA algorithms with the GSM-x instances datasets

Instances	|F|	Results	JA	(MJA)	(AJA)	Metaheuristics best results
GSM-93 (93)	9	Best	101	38	**33**	32
		Average	108.5	44.5	**34.9**	
		Worst	113	45	35	
		Std.	55.74	20.58	17.03	
GSM-93	13	Best	21	15	8	7
		Average	21.8	16.8	8.6	
		Worst	22	10	18	
		Std.	1.31	0.63	0.51	
GSM-246 (246)	21	Best	468	236	**90**	79
		Average	477.8	241.3	**91.4**	
		Worst	487	249	93	
		Std.	237.09	117.61	46.89	

TABLE 6.3. – Results obtained by the JA, MJA and AJA algorithms with the Test instances datasets

Instances	|F|	Results	JA	(MJA)	(AJA)	Metaheuristics best results
Test95 (95)	36	Best	25	13	**9**	8
		Average	27	15.37	**9.41**	
		Worst	29	17	10	
		Std.	1.6	1.58	0.51	
Test282 (282)	61	Best	470	176	**113**	51
		Average	490.6	181.33	**126.9**	
		Worst	504	185	163	
		Std.	237.09	117.61	46.89	

TABLE 6.4. – Results obtained by the JA, MJA and AJA algorithms with the instances datasets

Instances	|F|	Results	JA	(MJA)	(AJA)	Metaheuristics best results
P06-5	11	Best	245	140	137	133
		Average	247.87	144.23	137.26	
		Worst	248	145	138	
		Std.	3.81	2.61	0.45	
P06-3 (153)	31	Best	505	366	**118**	115
		Average	518.3	379.7	**120.1**	
		Worst	524	389	121	
		Std.	265.92	194.86	59.59	
P06b-5	31	Best	39	35	26	25
		Average	40.15	35.2	26.1	
		Worst	43	36	27	
		Std.	6.56	0.41	0.31	
P06b-3	31	Best	133	116	112	112
		Average	133.5	121	117	
		Worst	135	129	120	
		Std.	4.95	4.64	3.18	

TABLE 6.5. – Results obtained by the JA, MJA and AJA algorithms with the instances GSM2 datasets

Instances	|F|	Results	JA	(MJA)	(AJA)	Metaheuristics best results
GSM2-184	39	Best	5995	6334	**5878**	5250
		Average	6476.4	6424.34	**6170.71**	
		Worst	6836	6776	6526	
		Std.	306.73	281.50	191.47	
GSM2-227	29	Best	71386	67586	**66586**	57731
		Average	77426.91	68840.82	**67721**	
		Worst	79331	74423	69566	
		Std.	2264.35	3668.68	1096.28	
GSM2-272	34	Best	68665	63568	**62150**	53080
		Average	71766.9	74409.9	**65888.3**	
		Worst	77406	75405	67755	
		Std.	3164.63	4466.47	1477.70	

TABLE 6.6. – Results obtained by the JA, MJA and AJA algorithms with the instances r- datasets

Instances	|F|	Results	JA	(MJA)	(AJA)	Metaheuristics best results
1-1-50-75-30-2-50 (75)	5	Best	2055	1253	**1243**	1242
		Average	2286.25	1326.9	1246.2	
		Worst	2405	1407	1409	
		Std.	1117.6	648.5	637.19	
1-2-50-75-30-4-50 (75)	9	Best	1688	680	**673**	665
		Average	1726.8	703.25	**692.81**	
		Worst	1885	733	704	
		Std.	100.2	32.5	33.6	
1-3-50-75-30-0-50	7	Best	330	255	199	194
		Average	350.61	257.69	200.76	
		Worst	359	269	202	
		Std.	8.64	4.40	1.09	
1-4-50-75-30-2-1 (75)	6	Best	132	73	**70**	70
		Average	136.5	76.4	**70**	
		Worst	146	80	70	
		Std.	66.65	37.29	0	

Les meilleurs résultats sont toujours obtenus par la méthode AJA. La moyenne de la méthode AJA est également meilleure que les deux autres méthodes. L'écart-type de la méthode Jaya est également beaucoup plus grand, tandis que l'écart type des méthodes MJA et AJA sont plus petits.
Nous illustrons dans la Figure 6.1 les valeurs des moyennes de Jaya, MJA et AJA pour quelques instances. La figure 6.1 montre graphiquement que les algorithmes AJA et MJA ont produit des résultats supérieurs à Jaya pour toutes les instances considérées. Le graphique montre également que l'AJA et la MJA ont agi de la même manière dans certains cas et que l'AJA a obtenu de meilleurs résultats que la MJA pour d'autres. Nous présentons dans les graphiques de la figure 6.2 à la figure 6.5 les valeurs de la solution pour chaque instance de plus de dix exécutions pour les trois méthodes. Les graphiques montrent nettement l'écart entre les résultats obtenus par Jaya et les deux autres méthodes (MJA, AJA).

6.5.1. Analyse statistique

Nous avons voulu vérifier si les différences entre les solutions trouvées par les trois méthodes étaient significatives. A cet effet nous effectuons le test non paramétrique de Friedman pour déterminer si les algorithmes sont statistiquement différents.
Nous avons effectué le test de Friedman sur les jeux de données afin de détecter toute différences entre les trois algorithmes proposés. Nous avons effectué le test avec le R logiciel sous le niveau de signification alpha = 0,05.
Les résultats du test sont les suivants : Friedman chi-squared = 39.024, df = 2, p-value = 3.357e-09. Comme on peut le constater, le test montre la signification

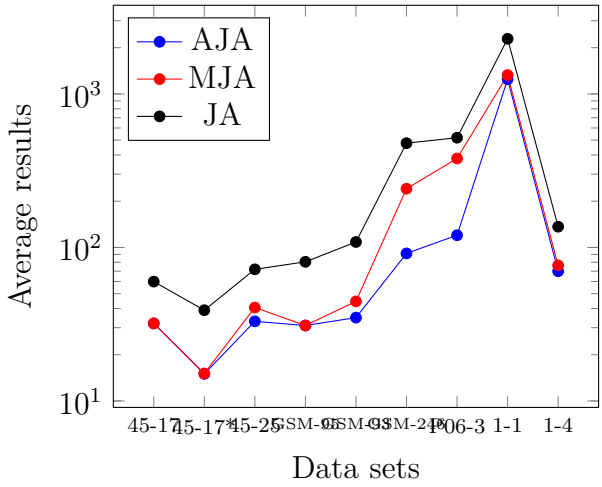


FIGURE 6.1. – Average results of the algorithms on 9 picked instances

(valeur p <0,05), de sorte que l'hypothèse nulle peut être rejetée car la valeur p est inférieure à 0,05. Les résultats sont donc différents. L'hypothèse nulle du test de Friedman étant rejetée, il est nécessaire de déterminer quels algorithmes diffèrent les uns des autres. À cette fin, nous avons effectué un test post-hoc de Nemenyi pour comparer différentes paires d'algorithmes. Les résultats du test post-hoc de Nemenyi sont résumés dans le tableau 6.7. Au vu de ces résultats, nous pouvons conclure que les algorithmes comparés sont statistiquement différents. En effet, on peut voir que le AJA diffère considérablement (p <0,05) de Jaya. En revanche, la différence entre AJA et MJA est limite significative. Il est de même pour la différence entre MJA et JA, la différence est aussi limite significative. Il est donc évident que les modifications apportées à l'algorithme Jaya ont eu un impact réel sur les performances de l'algorithme.

TABLE 6.7. – Nemenyi post hoc tests

Méthodes	JA	MJA
MJA	0.0058	–
AJA	2e-09	0.0058

6.6. Synthèse

Cette étude portait sur la résolution du problème MI-FAP à travers l'algorithme de Jaya. Nous concevons une adaptation de cet algorithme récemment développé appelé algorithme Jaya. De plus, nous proposons deux améliorations de l'algorithme Jaya. A ce titre, notre démarche porte tout d'abord sur une variante mémétique de Jaya et sur une variante adaptative de l'algorithme Jaya. L'adop-

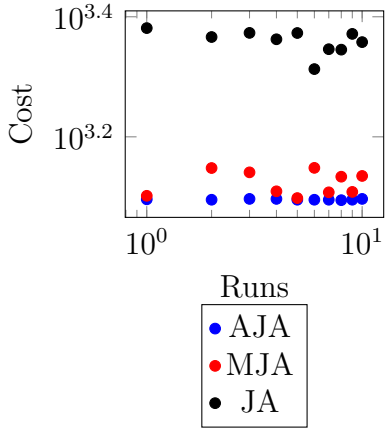


FIGURE 6.2. – Details of experiments on the instance 1-1-50-75-30-2-50

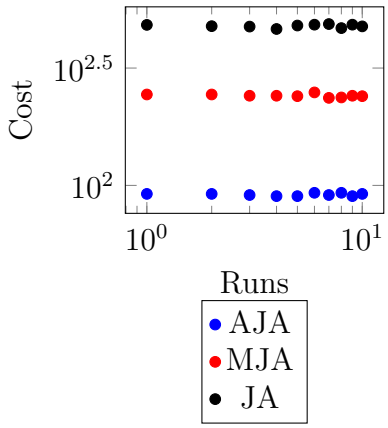


FIGURE 6.3. – Details of experiments on the instance GSM-246

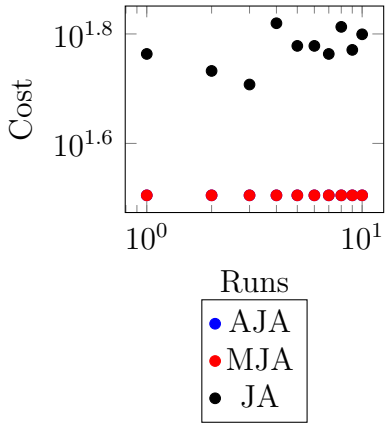


FIGURE 6.4. – Details of experiments on the instance 45-17 with 7 frequencies

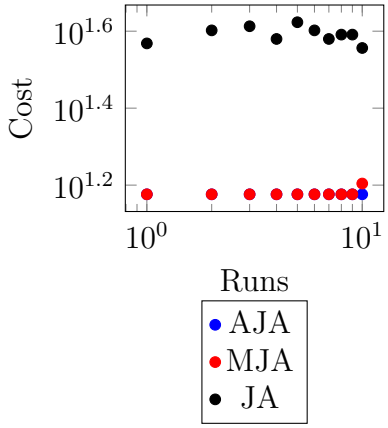


FIGURE 6.5. – Details of experiments on the instance 45-17 with 9 frequencies

tion et l'adaptation de l'algorithme Jaya au problème considéré et les deux améliorations proposées pour cet algorithme sont nos principales contributions dans cette partie. Les expériences montrent clairement que la méthode AJA donne de meilleurs résultats que les méthodes Jaya pure et MJA. Les résultats de l'AJA sont compétitifs et très encourageants. Par conséquent, nous pensons que l'approche AJA s'avère intéressante pour résoudre le MI-FAP. En ce qui concerne les travaux futurs, nous prévoyons d'étudier la possibilité d'utiliser une autre approche afin d'améliorer l'efficacité de la méthode JA.

7. Une approche hyper-heuristiques pour MI-FAP

Sommaire

7.1. Introduction

Pour résoudre le problème d'affectation de fréquences dans un réseau cellulaire nous nous sommes également intéressés aux hyper-heuristiques. Ces dernières ayant démontré leur puissance à résoudre des problèmes complexes. Dans ce chapitre, nous présentons l'essentiel de notre travail consacré aux hyper-heuristiques. Pour ce faire, nous structurons le chapitre comme suit : en premier lieu, nous définissons les principales caractéristiques d'une hyper-heuristique. En second lieu,nous introduirons les trois différentes hyper-heuristiques que nous avons proposés à savoir : l'hyper-heuristique à fonction de choix, l'hyper-heuristique aléatoire et une hyper-heuristique hybride. Ensuite nous présenterons et analyserons les résultats des expérimentations réalisées sur des benchmarks publics de diverses tailles. Enfin nous exposons nos conclusions sur l'approche hyper-heuristique pour résoudre le MI-FAP.

7.2. Les hyper-heuristiques

La difficulté de réglage des paramètres lors de l'application de méta-heuristiques pour résoudre un problème a conduit les chercheurs à la dure tâche de développer de nouvelles méthodes de recherche permettant d'opérer à un niveau d'abs-

traction plus élevé. L'objectif est donc d'aboutir au développement de méthodes de recherches plus générales, qui soient indépendantes du problème traité offrant ainsi plus de facilité d'adaptation aux différents problèmes considérés.
De ces motivations, sont nées les hyperheuristiques (hh). Ces dernières sont vues comme étant des méthodologies de recherche travaillant à un haut niveau d'abstraction plus élevé et d'indépendance par rapport au problème d'optimisation traité. Le terme hyper-heuristique a été utilisé pour la première fois en 1997 pour décrire un protocole la combinaison de nombreuses méthodes d'intelligence artificielle travaillant ensemble pour l'automatisation de la démonstration de théorème[14].Dans le cadre de l'optimisation combinatoire le terme hyperheuristique est défini dans [14] comme **heuristics to choose heuristics**, c'est-'à-dire des heuristiques pour le choix des heuristiques. En conséquence, l'espace de recherche de l'hyper-heuristique est alors centré sur l'heuristique plutôt que sur l'espace de recherche des solutions, contrairement aux méta-heuristiques. L'hyperheuristique travaille ainsi sur un niveau plus élevé, on parle alors d'heuristique de haut niveau. Cette dernière manipule des heuristiques de bas niveaux, dépendantes du problème, et a comme rôle de permettre le choix de l'heuristique adéquate parmi d'autres heuristiques existantes sur la base d'un système d'apprentissage. On trouve dans la littérature une autre variante d'hyperheuristiques. Celle-ci a pour rôle de générer des heuristiques. On parle alors de deux classes d'hyperheuristiques : les hyperheuristiques de génération et les hyperheuristiques de sélection.
Depuis leur parution, les hyper-heuristiques ont suscité l'intérêt de nombreux chercheurs et ont été exploitées avec succès sur différents problèmes d'optimisation tels que : Burke et al. en 2003 puis en 2007.[15, 16], Cowling et al. [22].
Notre travail consacré aux hyperheuristiques nous a mené à proposer trois hyperheuristiques de sélection. Les hyperheuristiques que nous avons proposé sont des méthodes de haut niveau, qui manipulent un ensemble d'heuristiques de bas niveaux, en se basant sur différents mécanismes lors de la sélection de l'heuristique de bas niveau à appliquer pour résoudre le MI-FAP. La première hyperheuristique se base sur une fonction de choix et nommée Hyperheuristique à fonction de choix, elle est notée (HFC). La deuxième choisit aléatoirement l'heuristique à appliquer et nommée Hyperheuristique aléatoire notée (HHA) et enfin la troisième nommée Stochastique Hyper-Heuristique (SHH) combine les deux précédentes méthodes de sélection pour choisir l'heuristique de bas niveau à utiliser. Les approches hyper-heuristiques proposées dans ce travail manipulent cinq heuristiques de bas niveau. Des expérimentations numériques sont réalisées sur différents benchmarks dans le but de tester et de prouver l'efficacité des approches proposées.
Nous consacrons la section suivante à la présentation des cinq heuristiques de bas niveaux utilisées par les trois hypehreuristiques que nous proposons pour le problème MI-FAP. Ensuite nous présenterons les trois hyperheuristiques.

7.3. Les heuristiques de bas niveau pour le MI-FAP

Les heuristiques de bas niveau manipulées par l'hyperheuristiques sont des méthodes, stratégies de recherche spécifiques au problème. Nous détaillons les heuristiques que nous avons développées spécifiquement au MI-FAP :

- L'heuristique h1 : h1 implémente une recherche locale de base et simple à mettre en œuvre. C'est une procédure itérative qui démarre d'une solution initiale générée aléatoirement X. A chaque itération, une solution voisine est obtenue en appliquant un mouvement. Pour le problème MI-FAP, le mouvement consiste à modifier la fréquence d'un (TRX). Le choix du (TRX) se fait aléatoirement parmi l'ensemble des ($TRXs$). Le choix de la nouvelle fréquence à affecter se fait de façon aléatoire. Le processus est répété un nombre de fois avec pour objectif l'amélioration de la solution.

- L'heuristique h2 : h2 est une recherche locale stochastique. Elle prend en entrée la meilleure solution trouvée, puis génère un voisin en appliquant un mouvement. Comme pour l'heuristique h1 le mouvement consiste à modifier la fréquence d'un (TRX). Pour h2 le mouvement s'effectue en utilisant une probabilité fixe $wp > 0$. La probabilité $wp > 0$ intervient lors du choix du (TRX) dont la fréquence sera modifiée. Le choix du (TRX) se fait selon les deux critères suivants :

  - Le premier critère consiste à sélectionner un (TRX) parmi l'ensemble des ($TRXs$) de manière aléatoire avec une probabilité $wp > 0$. La règle du mouvement ici est similaire à celle appliquée dans h1.

  - Le deuxième critère consiste à choisir, avec une probabilité $1 - wp$, le (TRX) ayant respecté le moins de contraintes, c'est- à -dire le TRX qui a le plus d'interférences et modifier sa fréquence actuelle par une fréquence lui permettant de réduire ses interférences.

- L'heuristique h3 : h3 reprend le principe d'un opérateur de mutation qui selon un paramètre de mutation noté (mrate) appliquera ou non, pour chaque (TRX) d'une solution X, une perturbation. C'est- à -dire que l'affectation de fréquence actuelle de chaque (TRX) sera modifiée ou non selon le paramètre mrate. La solution obtenue est ensuite passée comme paramètre d'entrée (solution de départ) à une recherche locale pour l'améliorer. La recherche locale appliquée est implémentée par l'heuristique (h1).

- L'heuristique h4 : h4 s'inspire de l'opérateur de croisement d'un algorithme génétique afin de créer une nouvelle solution. Pour la mise en œuvre de h4 nous proposons de combiner la meilleure solution avec la

moins bonne solution obtenues à l'appel de cette heuristique. Ensuite on applique un opérateur de mutation sur la nouvelle solution obtenue. L'opérateur de mutation est celui proposée dans l'heuristique h3. Suite à la phase de mutation, nous appliquons la recherche locale stochastique implémentée par l'heuristique h2 afin d'améliorer la solution obtenue.

- L'heuristique h5 : h5 s'inspire tout comme l'heuristique h4 de l'opérateur de croisement d'un algorithme génétique dans le but de créer une nouvelle solution. Pour la mettre en œuvre, nous proposons de combiner la meilleure solution avec une nouvelle solution générée aléatoirement. Ensuite on applique l'heuristique h3 à la solution obtenue. Cette solution est ensuite améliorée en utilisant la recherche locale stochastique implémentée par h2.

Nous sommes maintenant en mesure de de décrire les différentes hyperheuristiques que nous proposons pour résoudre le problème MI-FAP.

7.4. Hyperheuristique aléatoire pour le MI-FAP

Au cours de notre recherche consacrée aux traitements du MI-FAP par hyperheuristique nous avons pensé en premier lieu à une hyperheuristique des plus simples à implémenter. En effet la méthode utilisée pour la sélection des heuristiques n'est pas très élaborée. Elle sélectionnera l'heuristique de bas niveau à appliquer de façon aléatoire. Ceci donnera ainsi une chance à chaque heuristique d'être appelée. Comme il a été souvent mentionné, la force de l'aléatoire réside dans sa capacité à apporter de la diversité. L'hyperheuristique utilisant cette sélection aléatoire est illustrée dans l'algorithme 8.

7.5. Hyperheuristique à fonction de choix pour le MI-FAP

Dans un second temps, nous pensons à une hyperheuristique plus élaborée destinée à chercher davantage l'intensification que la diversité. Nous proposons alors une approche hyperheuristique basée sur une fonction de choix (HFC) pour sélectionner l'heuristique de bas niveau à appeler. Ainsi, nous adaptons la fonction de choix utilisée dans [22] pour le problème de scheduling au problème MI-FAP. HFC associe à chaque heuristique de bas niveau un score qui correspond à son efficacité. Ce score permettra de sélectionner l'heuristique la plus performante à chaque étape du processus de recherche.
Afin d'attribuer un score à chaque heuristique de bas niveau, la fonction de choix se base sur les éléments suivants :

Algorithm 8 L'hyper-heuristique à sélection aléatoire pour le MI-FAP.

Require: Une instance du MI-FAP, un ensemble d'heuristiques.
Ensure: meilleure solution S^*

```
1: générer une solution initiale S
2: Évaluer la qualité de la solution S
3: S*= S ; F*= F ; F, F* qualité des solutions S respec S* obtenues grâce à la
   formule (1).
4: while ( le critère d'arrêt non rencontré) do
5:    Sélection aléatoire de l'heurisitque h_i.
6:    Appliquer h_i sur S afin d'obtenir une nouvelle solution S' de qualité F'.
7:    if F' < F* then
8:       S*=S'.
9:       F*=F'.
10:   end if
11: end while
```

1. La performance récente de chaque heuristique (représentée par g_1).
2. La performance récente de paires d'heuristiques (représentée par g_2).
3. La récence du dernier appel de l'heuristique (représentée par g_3).

Ainsi nous avons :

$$\forall_i, g_1(h_i) = \sum_n \alpha^{n-1} \frac{I_n(h_i)}{T_n(h_i)}$$

$$\forall_i, g_2(h_{RL}, h_i) = \sum_n \beta^{n-1} \frac{I_n(h_{LL}, h_i))}{T_n(h_{RL}, h_i))}$$

$$\forall_i, g_3(h_i) = elapsedTime(h_i)$$

$$\alpha, \beta \in [0, 1], \gamma \in R$$

Avec :

- h_i : la $i^{ème}$ heuristique et h_{RL} la dernière heuristique lancée.
- α, β et γ sont des poids reflétant l'importance de chaque terme. Ils ont été fixés empiriquement.
- Les valeurs I_n et T_n donnent l'information sur la qualité de la solution et le temps CPU de la $i^{ième}$ heuristique.

Alors pour chaque heuristique de bas niveau le score sera définit comme suit :

$$\forall_i, score(h_i) = \alpha g_1(h_i) + \beta g_2(h_{RL}, h_i)) + \gamma g_3(h_i) \tag{7.1}$$

- g_1 est la performance récente de l'heuristique (h_i).
- g_2 est la performance récente de la paire d'heuristiques : l'heuristique (h_i) et celle la précédant (h_{RL}).
- g_3 mesure la récence du dernier appel de l'heuristique (h_i).

Les termes g_1 et g_2 sont les éléments qui permettent d'intensifier la recherche tandis que l'élément g_3 apporte de la diversification et cela en favorisant les heuristiques qui n'ont pas été choisies depuis un long moment.
Les détails de l'algorithme de l'hyper-heuristique basée sur une fonction de choix pour le MI-FAP sont présentés dans l'algorithme 9.

Algorithm 9 L'hyper-heuristique basée sur une fonction de choix pour le MI-FAP.

Require: Une instance du MI-FAP, un ensemble d'heuristiques, la fonction de choix, les paramètres α, β and γ
Ensure: meilleure solution S^*
1: Génerer une solution initiale S
2: Evaluer la qualité de la solution S
3: $S^*= S$; $F^*= F$; F, F^* qualité des solutions S respec S^* obtenues par la formule (1).
4: **while** (le critère d'arrêt non vérifié) **do**
5: Pour chaque heuristique h_i, calculer son score en utilisant la fonction de choix.
6: Sélection de l'heurisituqe h_i ayant le score le plus élevé.
7: Appliquer h_i sur S afin d'obtenir une nouvelle solution $S^{'}$ de qualité $F^{'}$.
8: **if** $F' < F^*$ **then**
9: $S^*=S'$.
10: $F^*=F'$.
11: **end if**
12: **end while**

Les termes g1 et g2 sont les éléments qui permettent d'intensifier la recherche tandis que l'élément g3 apporte de la diversification et cela en favorisant les heuristiques qui n'ont pas été choisies depuis un long moment. L'hyperheuristique basée sur cette fonction de choix est présentée dans l'algorithme 9.

7.6. L' Hyperheuristique stochastique pour le MI-FAP

La troisième hyper-heuristique que nous proposons combine les deux stratégies de sélections introduites précédemment : la sélection basée sur la fonction de

choix et la sélection aléatoire de l'heuristique à appeler. A chaque étape, le processus de recherche sélectionne selon une probabilité fixe $wp_1 > 0$ une des deux méthodes de sélections :

- Sélectionner aléatoirement l'heuristique à appliquer avec une probabilité fixe $wp_1 >$0.
- Sélectionner l'heuristique en utilisant la fonction de choix avec une probabilité $1 - wp_1$.

Les détails de l'algorithme de l'hyper-heuristique stochastique sont illustrés par l'algorithme 10.

Algorithm 10 La stochastique hyper-heuristique pour le MI-FAP.

Require: Une instance du MI-FAP, un ensemble d'heuristiques, la probabilité wp_1, la fonction de choix, α, β, γ.
Ensure: meilleure solution S^*

```
1: générer une solution initiale S
2: Evaluer la qualité de la solution S
3: S*= S ; F*= F ; F, F* qualité des solutions S respec S* obtenues grâce à la
   formule (1).
4: while ( le critère d'arrêt non rencontré) do
5:    r ← un nombre aléatoire entre 0 et 1 ;
6:    if (r < wp1) then
7:       hi=Une heuristique choisie aléatoirement ;
8:    else
9:       hi=Une heuristique choisie selon la fonction de choix ;
10:   end if
11:   Appliquer hi sur S afin d'obtenir une nouvelle solution S' de qualité F'.
12:   if F' < F* then
13:      S*=S'.
14:      F*=F'.
15:   end if
16: end while
```

7.7. Expérimentations et résultats numériques

Du fait de la nature non déterministe des algorithmes que nous proposons, des séries d'expériences de 20 tests ont été effectuées pour les différents benchmarks. Le critère d'arrêt des algorithmes est défini en fonction du temps CPU. Ce paramètre est fixé à 2400 secondes. Les résultats affichés dans les tableaux 7.1 à

7.6 correspondent aux valeurs moyennes (Average), la meilleure valeur (Best), la valeur la moins bonne (worst) de la fonction objectif, obtenus sur 20 exécutions pour chacun des algorithmes testés. La valeur (std) correspond à l'écart type entre les différentes exécutions. Les résultats en gras indiquent les meilleurs résultats obtenus en termes de meilleure valeur (Best) et valeur moyenne (average).

TABLE 7.1. – Results obtained by the CFH, RHH and SHH algorithms with the AC-x-y instances datasets

Instances	|F|	Results	CFH	(RHH)	(SHH)	Metaheuristics best results
AC-45-17	7	Best	45	42	32	32
		Average	46.75	42.35	**32**	
		Worst	47	46	32	
		Std.	1.25	1.05	0	
AC-45-17	9	Best	27	25	20	15
		Average	28.5	25.5	**22**	
		Worst	30	26	24	
		Std.	1.14	1.1	1.12	
AC-45-25	11	Best	38	36	33	33
		Average	38.2	35.5	**33**	
		Worst	39	35	33	
		Std.	1.26	0.4	0	
AC-95-9	6	Best	40	31	31	31
		Average	39.5	32.3	**31**	
		Worst	41	33	31	
		Std.	1.12	0.67	0	
AC-95-17	21	Best	40	39	17	10
		Average	43.25	39.1	**17.63**	
		Worst	45	40	19	
		Std.	1.86	1.20	1.14	

Les résultats expérimentaux présentés dans les tableaux 7.1 à 7.6 résument les résultats obtenus. Nous constatons des résultats présentés dans le tableau 7.1 que SHH obtient de meilleur résultats pour toutes les instances testées. Elle réussit également à trouver le même résultat que l'état de l'art pour les instances AC-45-17 (7), AC-45-25 et pour l'instance AC-95-9. Les méthodes CFH et RHH enregistrent des résultats de bien moins bonne qualité que ceux de l'état de l'art. En comparant entre ces deux méthodes -pour ces intances- on observe que la RHH atteint de meilleurs résultats que la CFH. Les résultats obtenus sur les instances GSM-x, présentés dans le tableau 7.2, et les instances Test, présentés dans le tableau 7.3, montrent l'efficacité de la méthode SHH par rapport aux deux autres méthodes. Les résultats obtenus par la CFH et la RHH sont comparables. Les résultats enregistrés par la méthode SHH sont plus proches de ceux de l'état de l'art en comparaison avec ceux de CFH et RHH.

TABLE 7.2. – Results obtained by the CFH, RHH and SHH algorithms with the GSM-x instances datasets

Instances	|F|	Results	CFH	(RHH)	SHH	Metaheuristics best results
GSM-93	9	Best	38	38	37	32
		Average	41.25	41.2	**39.18**	
		Worst	45	42	40	
		Std.	2.52	1.03	0.89	
GSM-93	13	Best	20	18	11	7
		Average	23.8	23.28	**13**	
		Worst	25	25	15	
		Std.	1.31	0.63	0.51	
GSM-246	21	Best	95	94	85	79
		Average	96.86	97.72	**86.78**	
		Worst	99	99	89	
		Std.	1.03	2.32	1.71	

TABLE 7.3. – Results obtained by the CFH, RHH and SHH algorithms with the Test instances datasets

Instances	|F|	Results	CFH	RHH	(SHH)	Metaheuristics best results
Test282	61	Best	95	93	82	51
		Average	100.2	97.63	**84.5**	
		Worst	108	109	88	
		Std.	3.6	5.18	2.26	
Test95	36	Best	15	16	15	8
		Average	16.61	16.7	**15**	
		Worst	17	18	15	
		Std.	1.26	1.08	0	

TABLE 7.4. – Results obtained by the CFH, RHH and SHH algorithms with the instances P0b-x datasets

Instances	|F|	Results	CFH	RHH	SHH	Metaheuristics best results
P06-5	11	Best	150	150	139	133
		Average	151.87	151.23	**139.26**	
		Worst	154	153	140	
		Std.	3.81	2.61	0.45	
P06-3	31	Best	124	124	117	115
		Average	127.5	126.5	**118.1**	
		Worst	128	126	120	
		Std.	2.19	0.00	0.31	
P06b-5	31	Best	39	38	26	25
		Average	40.25	38.2	**26.1**	
		Worst	44	40	29	
		Std.	6.56	1.41	0.31	
P06b-3 3	1	Best	128	126	115	112
		Average	125.5	121.4	**117**	
		Worst	130	128	120	
		Std.	4.95	4.64	3.18	

TABLE 7.5. – Results obtained by the CFH, RHH and SHH algorithms with the instances GSM2 datasets

Instances	|F|	Results	CFH	RHH	SHH	Metaheuristics best results
GSM2-184	39	Best	5994	6234	5898	5250
		Average	6475.4	6524.34	**6180.71**	
		Worst	6835	6786	6826	
		Std.	306.73	281.50	191.47	
GSM2-227	29	Best	71586	68586	67586	57731
		Average	77626.91	69840.82	**68721**	
		Worst	79531	75423	70566	
		Std.	2264.35	3668.68	1096.28	
GSM2-272	34	Best	98664	95568	85150	53080
		Average	891760.9	94409.9	**87488.3**	
		Worst	99806	95405	89655	
		Std.	3164.63	4466.47	1477.70	

TABLE 7.6. – Results obtained by the CFH, RHH and SHH algorithms with the instances r- datasets

Instances	|F|	Results	CFH	RHH	SHH	Metaheuristics best results
1-1-50-75-30-2-50	5	Best	1478	1256	1277	1242
		Average	1487.18	1290	**1278.44**	
		Worst	1544	1389	1285	
		Std.	19.25	38.33	9.12	
1-2-50-75-30-4-50	9	Best	713	682	670	665
		Average	751.61	695.69	**676.18**	
		Worst	781	764	679	
		Std.	27.78	28.77	2.89	
1-3-50-75-30-0-50	7	Best	218	216	202	194
		Average	220.61	219.69	**200.5**	
		Worst	230	228	205	
		Std.	8.64	13.40	1.59	
1-4-50-75-30-2-1	6	Best	85	85	72	70
		Average	83.2	85	**74.2**	
		Worst	86	87	75	
		Std.	2.53	1.04	2.35	

Pour les instances P0b-x dont les résultats sont résumés dans le tableau 7.4 on observe toujours la supériorité de la méthode SHH. On observe également que pour les instances P06-3 et P06b-5 la méthode SHH obtient des résultats comparables à ceux de l'état de l'art. Les résultats de CFH et RHH restent de moins bonne qualité. En moyenne les résultats entre ces deux méthodes sont comparables.
Les résultats présentés dans le tableau 7.5 montrent que SHH atteint de meilleur résultats. Les résultats indiquent également qu'aucune des méthodes ne parvient à atteindre les résultats de l'état de l'art. Les résultats présentés dans le tableau 7.6 montrent que SHH donne de meilleur résultats par rapport à ceux enregistrés par CFH et RHH. Les résultats indiquent également que pour l'instance 1-4-50-75-30-2-1 la méthode SHH enregistre des résultats proches de l'état de l'art.

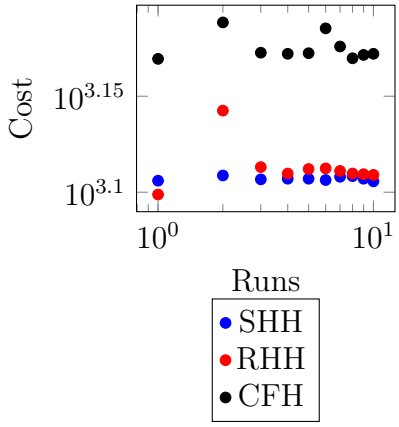


FIGURE 7.1. – Details of experiments on the instance 1-1-50-75-30-2-50

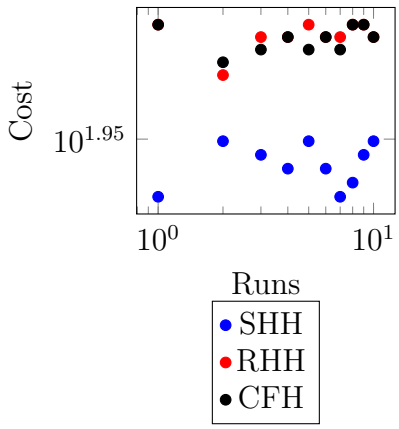


FIGURE 7.2. – Details of experiments on the instance GSM-246

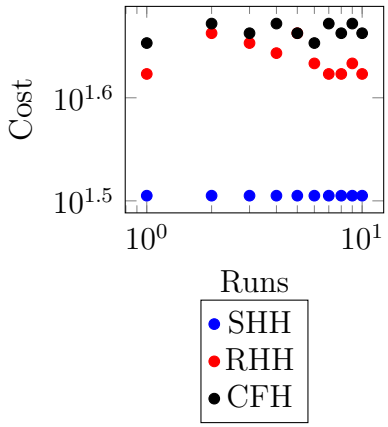


FIGURE 7.3. – Details of experiments on the instance AC45-17(7)

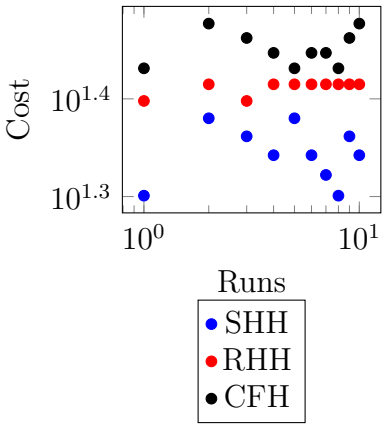


FIGURE 7.4. – Details of experiments on the instance AC45-17(9)

Nous pouvons voir que l'hyper-heuristique stochastique (SHH) est bien meilleure que les deux hyper-heuristiques CFH et RHH pour toutes les instances testées. On constate également que les résultats entre CFH et RHH sont dans la plus part des cas comparables. L'efficacité de la méthode SHH est due à la bonne combinaison entre le la fonction de choix et le caractère aléatoire qui permettent de sélectionner efficacement l'heuristiques de bas niveau. Elle est capable d'explorer l'espace de recherche et localiser des solutions de bonnes qualités. Les résultats de SHH sont donc toujours de meilleure qualité que CFH et RHH en termes de moyenne et de best solution pour tous les groupes d'instances. Cependant, nous observons qu'aucune des trois méthodes proposées n'a pu atteindre les résultats trouvés dans la littérature.

7.7.1. Analyse statistique

Nous avons voulu vérifier si les différences entre les solutions dégagées par les trois méthodes étaient significatives. A cet effet nous effectuons le test non paramétrique de Friedman pour déterminer si les algorithmes sont statistiquement différents.
Les résultats du test sont les suivants : Friedman chi-squared = 43.28, df = 2, p-value = 3.998e-10.
Comme on peut le constater, le test montre la signification (valeur p <0,05), de sorte que l'hypothèse nulle peut être rejetée car la valeur p est inférieure à 0,05. Les résultats sont donc différents. L'hypothèse nulle du test de Friedman étant rejetée, il est nécessaire de déterminer les algorithmes qui diffèrent les uns des autres. À cette fin, nous avons effectué un test post-hoc de Nemenyi afin de comparer différentes paires d'algorithmes. Les résultats du test post-hoc de Nemenyi sont résumés dans le tableau 7.7. Au vu de ces résultats, nous pouvons conclure que les algorithmes comparés sont statistiquement différents. En effet, on peut voir que le SHH diffère considérablement (p <0,05) du CFH et de RHH et que ce dernier diffère de CFH (p <0,05).

TABLE 7.7. – Nemenyi post hoc tests

Méthodes	CFH	RHH
RHH	0.04282	–
SHH	2.3e-10	0.00012

7.8. Synthèse

Le travail présenté dans ce chapitre a pour objectif d'introduire trois approches basées sur l'approche hyper-heuristique pour résoudre le problème d'affectation de fréquence dans un réseau cellulaire. Nous avons présenté, dans un premier temps trois variantes d'hyper-heuristique : une hyper-heuristique basée sur une fonction de choix, une hyper-heuristique à sélection aléatoire et une hyperheuristique stochastique. Les hyper-heuristiques proposées manipulent un ensemble d'heuristiques de bas niveaux. La différence entre les hyper-heuristiques développées réside dans le mécanisme de sélection de l'heuristique à appeler. Nous avons implémenté puis testé nos méthodes sur différentes instances du problème d'affectation de fréquences.
Les résultats expérimentaux indiquent de manière claire la supériorité de l'hyperheuristique SHH. Néanmoins, les résultats montrent que la performance des hyper-heuristiques proposées ne parvient pas à atteindre les résultats obtenus dans la littérature. Pour traiter ces limites, il serait intéressant d'engager une réflexion sur d'autres heuristiques de bas niveau et une étude plus approfondie des paramètres des algorithmes proposés.

Troisième partie

Conclusion générale

Conclusion générale

L'affectation de fréquences est une étape cruciale dans le design d'un réseau de télécommunication, décrite comme un problème d'optimisation combinatoire qui revient à trouver un plan de fréquences optimal de façon à satisfaire la demande en trafic et tout en minimisant les interférences. En pratique, les opérateurs de téléphonie mobile disposent d'un nombre limité de fréquences radio qui représentent une ressource rare et coûteuse. Ils gèrent également un grand nombre d'émetteurs/récepteurs ($TRXs$) auxquels il faut affecter des fréquences pour assurer le service. Afin de gérer cette contrainte, les opérateurs font appel au principe de réutilisation de fréquences. Ce dernier n'a pas que des avantages car il a pour effet d'engendrer des interférences particulièrement lorsque des ($TRXs$) proches géographiquement utilisent la même fréquence ou des fréquences adjacentes. Le problème revient alors à minimiser les interférences lors de l'affectation des fréquences (the minimum interference frequency assignment problme MI-FAP).
Dans cette thèse nous avons adapté trois métaheuristiques, à savoir la recherche à voisinage variable, la recherche harmonique et l'algorithme de jaya. Nous avons également proposé une approche hyperheuristiques.
Les premiers résultats obtenus par l'adaptation de VNS n'ont pas atteint les résultats trouvés dans la littérature. Nous avons alors proposé certaines modifications à l'algorithme de base suite à nos premières observations. Nous avons utilisé dans un premier temps une VNS itérée pour pallier le problème de stagnation que rencontre VNS. Ainsi, l'optimal local est constamment perturbé. Les résultats ont été améliorés par rapport à ceux de VNS. Cependant les résultats restaient toujours de moindre qualité comparés à ceux trouvés dans la littérature. Nous avons alors pensé à un algorithme basé sur VNS un peu plus sophistiqué en proposant une VNS adaptative. Cet algorithme se base sur l'état de la recherche pour faire appel à différentes procédures. Dans cet algorithme, il est également fait appel à un concept issu des machines learning(OPBL). Les résultats enregistrés par le biais de cette méthode sont largement plus encourageants que la VNS. En effet, l'algorithme a réussi dans de nombreux cas à atteindre les résultats de l'état de l'art, à s'en rapprocher dans les autres cas. Néanmoins, les performances, pour les instances de grandes tailles, même si elles ont été améliorées, demeurent éloignées du résultat obtenu par l'état de l'art.
Les deuxièmes résultats concernent ceux obtenus par l'adaptation de la recherche harmonique. Au cours de nos expérimentations nous avons remarqué que les solutions s'améliorent très lentement et stagnent assez vite dans un mauvais opti-

mum local sans pouvoir s'en échapper. Pour remédier à cela et tenter d'atteindre des solutions plus intéressantes, nous avons choisi d'apporter une modification à HS de base. Ainsi, lors du processus d'improvisation d'HS,au lieu d'improviser une seule solution, il en improvise plusieurs. Cette capacité a permis d'améliorer les résultats et d'intensifier la recherche autour des solutions existantes de la population courante. Toutefois, les résultats restent de moindre qualité par rapport à ceux de l'état de l'art. Nous avons alors pensé à adopter une autre démarche en exploitant la force du memetic et de l'OPBL. En fonction des performances de HS, le processus tentera alors une amélioration par recherche locale ou un apport de diversité par l'OPBL. Guidé toujours par une optique d'amélioration, nous avons axé notre travail sur une collaboration entre HS et l'évolution différentielle. L'algorithme collaboratif met ainsi à jour la population en alternant entre HS et DE. Les différents résultats obtenus par les variantes proposées de HS peuvent être considérés comme encourageants dans la mesure où nous avons réussi à atteindre certains résultats de l'état de l'art.

Nous avons également étudié les possibilités offertes par un nouvel algorithme en l'occurrence l'algorithme de Jaya. Nous avons en premier lieu étudier et adapter cet algorithme au problème MI-FAP et également proposé deux variantes de cet algorithme à savoir Memetic Jaya, et Adaptative Jaya. Le premier est tout simplement un algorithme memetic qui combine entre Jaya et une recherche locale. Le second est un alogrithme adaptatif qui selon les performances de la recherche va adapter le comportement de l'algorithme. Les résultats obtenus par les deux améliorations proposées sont très encourageants.

Enfin, nous avons en outre étudié l'approche hyperheuristique. Au cours de nos investigations nous avons opté pour trois stratégies qui sont une hypeheuristique aléatoire, une hyperheuristique à fonction de choix et une stochastique hypeheuristique qui combinera les deux méthodes à savoir l'hypeheuristique aléatoire et l'hypeheuristique à fonction de choix. Nos résultats obtenus suite aux différentes expérimentations conduisent à la conclusion de la supériorité de la méthode alliant les deux stratégies de sélections.

En conclusion, nous pouvons dire que les changements que nous avons apportés aux différentes méthodes étudiées au cours de ces années de recherche ont été fructueux. Nous avons réussi à améliorer les résultats enregistrés par chacune de ces méthodes. Nous avons également réussi à atteindre les résultats mentionnés dans la littérature pour certaines instances et de nous en rapprocher grandement pour certaines instances et un peu moins pour d'autres. Néanmoins, nous faisons toujours face à certaines limitations qui empêchent nos méthodes de traiter efficacement les instances de grandes tailles du MI-FAP.

Comme perspective, il serait intéressant de pousser nos investigations sur les différentes possibilités, de faire collaborer plusieurs approches pour trouver une méthode efficace aux instances de grandes tailles en exploitant toutes les capacités et les innovations apportées par l'intelligence artificielle.

Quatrième partie

Bibliographie

Bibliographie

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

[2] E. Aarts and J. Korst. Simulated annealing and boltzmann machines. 1988.

[3] J. Alami and A. El Imrani. Using cultural algorithm for the fixed-spectrum frequency assignment problem. *Journal of Mobile Communication*, 2(1) :1–9, 2008.

[4] G. K. Audhya, K. Sinha, K. Mandal, R. Dattagupta, S. C. Ghosh, and B. P. Sinha. A new approach to fast near-optimal channel assignment in cellular mobile networks. *IEEE Transactions on Mobile Computing*, 12(9) :1814–1827, 2013.

[5] B. Autin. Les métaheuristiques en optimisation combinatoire. *Mémoire pour l'obtention l'examen probatoire en informatique, Conservatoire Nationales des Arts et des Métiers Paris*, 2006.

[6] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA journal on computing*, 6(2) :126–140, 1994.

[7] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2) :154–160, 1994.

[8] D. Beckmann and U. Killat. Frequency planning with respect to interference minimization in cellular radio networks. *Rapport technique, COST*, 259, 1999.

[9] L. Belhoul. *Résolution de problèmes d'optimisation combinatoire mono et multi-objectifs par énumération ordonnée*. PhD thesis, Université Paris Dauphine-Paris IX, 2014.

[10] U. Benlic and J.-K. Hao. Breakout local search for the max-cutproblem. *Engineering Applications of Artificial Intelligence*, 26(3) :1162–1173, 2013.

[11] T. Blickle and L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4) :361–394, 1996.

[12] A. Bouju, J. Boyce, C. Dimitropoulos, G. Vom Scheidt, and J. Taylor. Tabu search for the radio links frequency assignment problem. *Applied Decision Technologies (ADT'95)(London)*, 1995.

[13] I. Boussaid. *Perfectionnement de métaheuristiques pour l'optimisation continue*. PhD thesis, Paris Est, 2013.

[14] E. K. Burke, G. Kendall, et al. *Search methodologies*. Springer, 2005.

[15] E. K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of heuristics*, 9(6) :451–470, 2003.

[16] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1) :177–192, 2007.

[17] D. Castelino, S. Hurley, and N. Stephens. A tabu search algorithm for frequency assignment. *Annals of Operations Research*, 63(2) :301–319, 1996.

[18] G. Cheng, W. Liu, Y. Li, and W. Cheng. Joint on-demand routing and spectrum assignment in cognitive radio networks. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 6499–6503. IEEE, 2007.

[19] Y. Collette and P. Siarry. *Optimisation multiobjectif : Algorithmes*. Editions Eyrolles, 2011.

[20] D. Costa. On the use of some known methods fort-colorings of graphs. *Annals of Operations Research*, 41(4) :343–358, 1993.

[21] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the operational research society*, 48(3) :295–305, 1997.

[22] P. Cowling, G. Kendall, and E. Soubeiga. Hyperheuristics : A robust optimisation method applied to nurse scheduling. In *International Conference on Parallel Problem Solving from Nature*, pages 851–860. Springer, 2002.

[23] L. Davis. Handbook of genetic algorithms. 1991.

[24] C. Demoulin and M. Van Droogenbroeck. Principes de base du fonctionnement du réseau gsm. *Revue de l'AIM*, 4 :3–18, 2004.

[25] M. Dorigo and C. Blum. Ant colony optimization theory : A survey. *Theoretical computer science*, 344(2-3) :243–278, 2005.

[26] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8) :851–871, 2000.

[27] M. Duque-Antón, D. Kunz, and B. Ruber. Channel assignment for cellular radio using simulated annealing. *IEEE Transactions on Vehicular Technology*, 42(1) :14–21, 1993.

[28] K. Dyall, I. Grant, C. Johnson, F. Parpia, and E. Plummer. Grasp : A general-purpose relativistic atomic structure program. *computer physics communications*, 55(3) :425–456, 1989.

[29] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[30] A. Eisenblätter. The semidefinite relaxation of the k-partition polytope is strong. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 273–290. Springer, 2002.

[31] M. Ergezer, D. Simon, and D. Du. Oppositional biogeography-based optimization. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 1009–1014. IEEE, 2009.

[32] I. Flood and S. M. Allen. The fixed links frequency assignment problem with equipment selection. *Wireless personal communications*, 71(1) :181–194, 2013.

[33] M. R. Garey and D. S. Johnson. Computers and intractability : A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed. *Computers and Intractability*, 340, 1979.

[34] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm : harmony search. *simulation*, 76(2) :60–68, 2001.

[35] F. Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3) :190–206, 1989.

[36] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier, 1991.

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

[38] S. Hanafi, J. Lazić, N. Mladenović, C. Wilbaut, and I. Crévits. New hybrid matheuristics for solving the multidimensional knapsack problem. In *International Workshop on Hybrid Metaheuristics*, pages 118–132. Springer, 2010.

[39] P. Hansen and N. Mladenović. Variable neighborhood search : Principles and applications. *European journal of operational research*, 130(3) :449–467, 2001.

[40] J.-K. Hao and R. Dorne. Study of genetic search for the frequency assignment problem. In *European Conference on Artificial Evolution*, pages 333–344. Springer, 1995.

[41] J.-K. Hao, P. Galinier, and M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 13(2) :283–324, 1999.

[42] J.-K. Hao and L. Perrier. Tabu search for the frequency assignment problem in cellular radio networks. *European Journal of Operational Research*, 1999.

[43] A. Hertz, D. Schindl, and N. Zufferey. Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. *4or*, 3(2) :139–161, 2005.

[44] F. S. Hillier and G. J. Lieberman. Introduction to operations research. Technical report, Holden-Day San Francisco, 1967.

[45] M. Hollander and D. A. Wolfe. Nonparametric statistical methods. 1999.

[46] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598) :671–680, 1983.

[47] D. Kunz. Channel assignment for cellular radio using neural networks. *IEEE Transactions on Vehicular Technology*, 40(1) :188–193, 1991.

[48] X. Lai and J.-K. Hao. Path relinking for the fixed spectrum frequency assignment problem. *Expert Systems with Applications*, 42(10) :4755–4767, 2015.

[49] R. Leese and S. Hurley. *Methods and algorithms for radio channel assignment*, volume 23. Oxford University Press on Demand, 2002.

[50] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2) :498–516, 1973.

[51] F. Luna, C. Blum, E. Alba, and A. J. Nebro. Aco vs eas for solving a real-world frequency assignment problem in gsm networks. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 94–101. ACM, 2007.

[52] F. Luna, C. Estébanez, C. León, J. M. Chaves-González, A. J. Nebro, R. Aler, C. Segura, M. A. Vega-Rodríguez, E. Alba, J. M. Valls, et al. Optimization algorithms for large-scale real-world instances of the frequency assignment problem. *Soft Computing*, 15(5) :975–990, 2011.

[53] X. Ma, F. Liu, Y. Qi, M. Gong, M. Yin, L. Li, L. Jiao, and J. Wu. Moea/d with opposition-based learning for multiobjective optimization problem. *Neurocomputing*, 146 :48–64, 2014.

[54] H. Mabed, A. Caminada, and J.-K. Hao. Genetic tabu search for robust fixed channel assignment under dynamic traffic data. *Computational optimization and applications*, 50(3) :483–506, 2011.

[55] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6) :1087–1092, 1953.

[56] B. Metzger. Spectrum management technique presented at 38th national orsa meeting. *Detroit, MI (Fall 1970)*, 1970.

[57] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & operations research*, 24(11) :1097–1100, 1997.

[58] R. Montemanni, J. N. Moon, and D. H. Smith. An improved tabu search algorithm for the fixed-spectrum frequency-assignment problem. *IEEE transactions on vehicular technology*, 52(4) :891–901, 2003.

[59] R. Montemanni, D. Smith, and S. M. Allen. An improved algorithm to determine lower bounds for the fixed spectrum frequency assignment problem. *European Journal of Operational Research*, 156(3) :736–751, 2004.

[60] R. Montemanni and D. H. Smith. Heuristic manipulation, tabu search and frequency assignment. *Computers & Operations Research*, 37(3) :543–551, 2010.

[61] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4) :308–313, 1965.

[62] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 1998.

[63] K. V. Price, R. M. Storn, and J. A. Lampinen. The differential evolution algorithm. *Differential evolution : a practical approach to global optimization*, pages 37–134, 2005.

[64] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama. Opposition-based differential evolution for optimization of noisy problems. In *IEEE congress on evolutionary computation*, pages 1865–1872, 2006.

[65] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama. Opposition-based differential evolution. *IEEE Transactions on Evolutionary computation*, 12(1) :64–79, 2008.

[66] R. V. Rao and G. Waghmare. A new optimization algorithm for solving complex constrained design optimization problems. *Engineering Optimization*, 49(1) :60–83, 2017.

[67] M. Sakarovitch. *Programmation discrète : optimisation combinatoire ; méthodes mathématiques et algorithmiques*. Hermann, 1984.

[68] C. Segura, G. Miranda, and C. León. Parallel hyperheuristics for the frequency assignment problem. *Memetic Computing*, 3(1) :33–49, 2011.

[69] H. R. Tizhoosh. Opposition-based learning : a new scheme for machine intelligence. In *Computational intelligence for modelling, control and automation, 2005 and international conference on intelligent agents, web technologies and internet commerce, international conference on*, volume 1, pages 695–701. IEEE, 2005.

[70] S. Voß, S. Martello, I. H. Osman, and C. Roucairol. *Meta-heuristics : Advances and trends in local search paradigms for optimization*. Springer Science & Business Media, 2012.

[71] H. Wang, W. Wang, and H. Sun. Firefly algorithm with generalised opposition-based learning. *International Journal of Wireless and Mobile Computing*, 9(4) :370–376, 2015.

[72] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca. Enhancing particle swarm optimization using generalized opposition-based learning. *Information Sciences*, 181(20) :4699–4714, 2011.

[73] B. Weinberg, V. Bachelet, and E.-G. Talbi. A co-evolutionist meta-heuristic for the assignment of the frequencies in cellular networks. In *Workshops on Applications of Evolutionary Computation*, pages 140–149. Springer, 2001.

[74] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao. A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*, 29 :1–12, 2014.