
*ACADÉMIE D'AIX-MARSEILLE*
*UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE*

---

# THÈSE

## Recherche exacte et approchée en optimisation combinatoire : schémas d'intégration et applications

Présentée et soutenue publiquement le 20 mai 2005 pour obtenir le grade de Docteur en Sciences de l'Université d'Avignon et des Pays de Vaucluse

**SPÉCIALITÉ :** INFORMATIQUE

par Mireille Palpant

**Composition du jury :**

| | | | |
|---|---|---|---|
| M | Van-Dat Cung | PR, GILCO, Grenoble | Rapporteurs |
| M | Thierry Mautor | MdC HDR, PRISM, Versailles | |
| M | Abderrahim Benslimane | PR, LIA, Avignon | Examinateurs |
| M | Francis Sourd | CR, LIP6, Paris | |
| M | Michel Vasquez | HDR, LGI2P, Nîmes | |
| M | Christian Artigues | MdC HDR, LIA, Avignon | Directeurs de thèse |
| M | Philippe Michelon | PR, LIA, Avignon | |

---


Laboratoire Informatique d'Avignon

# Remerciements

Cela fait des heures à présent que je me suis attelée à ce difficile travail. Qui aurait cru que l'évidence soit si difficile à exprimer ?

Mais puisqu'il faut un commencement, je crois que cette place vous est dévolue d'office : mes chers parents, recevez toute la gratitude que j'ai parfois été bien avare de montrer. Pour vos encouragements et vos sacrifices, merci du fond du cœur. Merci également au reste de ma famille, et en particulier à mes grands-parents. Mamie, j'espère qu'aujourd'hui tu es fière de moi.

Si l'on suit le cours du temps, il est encore une personne sans qui je ne serai pas là. Victor Chepoi, pour la qualité de votre enseignement et pour m'avoir donné goût à ce que je fais aujourd'hui, je vous exprime ma plus profonde reconnaissance. Vous êtes à l'origine de bien des choses, sans même le savoir, probablement.

Un grand merci également à Christian Artigues, co-encadrant de cette thèse, mais ami avant tout. Je tiens tout particulièrement à souligner la qualité de ton encadrement, tes compétences scientifiques remarquables (allez, fais pas ton modeste :))), ainsi que tes qualités humaines - et surtout ta patience exemplaire ! ! ! Désolée si je t'ai fait avoir des cheveux blancs avant l'heure ;)

Je voudrais également saluer l'ensemble des membres du jury et, en particulier, Van-Dat Cung et Thierry Mautor, rapporteurs de cette thèse, pour avoir eu la patience de lire ce manuscrit jusqu'au bout. J'ai bien conscience de la quantité de travail non négligeable que cela constitue. Merci encore ! ! !

Je remercie également Thierry Defaix et Michel Vasquez pour m'avoir offert des sujets d'étude passionnants et enrichissants, et pour leur sympathie.

Je ne voudrais pas oublier non plus les membres du LIA et de l'IUP, dont certains sont devenus de véritables amis (you know who you are...), que je salue dans leur ensemble pour avoir contribué aux conditions de travail très chaleureuses. Je vais regretter les après-midi chômés pour des parties de pétanque ! ! ! Même si je ne gagne pas souvent ;o)

En vrac, et de façon non exhaustive : Catherine (désolée pour la musique ! ! !), Cristian (à moi le

Chili !!!), Sophie, Dominique, Olivier (tous !!!), les RAPeurs (beurk...) et autres RALeurs (j'aurais bien pû faire partie de cette équipe finalement :))), Christian, Jens, Thierry, Nico, et tous ceux que ma flemmardise m'empêche ici de nommer...

Et finalement merci à l'ensemble des mes amis, en particulier Delphine, Elina, Jérôme, Mig (à quand l'Everest ? :))), MJ et Eric (merci d'avoir eu pitié d'une pauvre SDF ;)), les fous du fofo Epica, et notamment Didier et Alain (chuis fane, ©Sarah).

Et mon bibou...

*À la vie...*

# Sommaire

# Table des figures

# Liste des tableaux

# Introduction

La recherche opérationnelle s'attache à étudier des problèmes dont la résolution, de par leur nature hautement *combinatoire*, constitue un véritable challenge. Il s'agit alors de trouver une *affectation* de valeurs à un certain nombre de *variables* tout en respectant un ensemble de *contraintes* donné. Les formalismes utilisés peuvent varier suivant le problème considéré et l'approche envisagée pour le résoudre.

Ainsi, les techniques de *programmation par contraintes* (PPC) sont-elles particulièrement adaptées pour étudier la *réalisabilité* d'un problème de satisfaction de contraintes, tandis que la *programmation linéaire en nombres entiers* (PLNE) s'inscrit davantage dans le cadre de la recherche d'un extremum d'une fonction linéaire. Cependant, l'une ou l'autre des approches, voire leur utilisation conjointe, peut être indifféremment adaptée pour modéliser et résoudre un problème de *décision* ou d'*optimisation*. En outre, elles partagent une procédure de résolution commune qui consiste en l'*énumération implicite* de l'ensemble des solutions du problème. Il s'agit alors de parcourir l'*espace de recherche* et d'en extraire une solution *admissible* (problèmes de satisfaction) ou *optimale* (problèmes d'optimisation) ou de prouver qu'il n'en existe pas. Le schéma classique consiste en une *recherche arborescente* qui évalue à chaque nœud la *configuration* (ou *solution partielle*) courante et l'étend si possible en affectant une valeur à une variable non encore instanciée. Ce processus s'avère néanmoins peu satisfaisant dès lors que le problème atteint une taille importante, en regard du temps d'exécution dispensé.

C'est pourquoi la résolution de nombreux problèmes fait souvent appel à des méthodes *incomplètes*, dont la gamme s'étend du simple algorithme glouton à des méthodes faisant intervenir des mécanismes beaucoup plus élaborés (stratégies évolutionnistes, recherche locale...). Le but avoué consiste alors à trouver un *compromis* entre la qualité des solutions et le temps passé à les chercher. Ainsi, il est possible d'introduire une certaine *souplesse* dans la gestion des objectifs, ce que ne permet pas une méthode complète. Cependant, le choix de cette dernière demeure le plus approprié lorsqu'il s'agit de résoudre des problèmes plus restreints ou qu'il est possible d'élaguer efficacement l'espace de recherche.

Cette constatation constitue le point de départ du travail présenté dans ce mémoire. Plus précisément, nous nous intéressons aux possibilités d'*hybridation* entre les deux approches afin de pouvoir tirer avantage de chacune d'elles : *systématicité* et *optimalité* de la résolution exacte, caractère moins *déterministe* et *rapidité* de la composante heuristique. Dans l'objectif de résoudre des problèmes NP-difficiles de taille relativement importante, nous nous intéressons exclusivement à la conception de méthodes incomplètes basées sur ces hybridations.

Nous avons pour cela divisé notre étude en deux axes distincts, dont les grandes lignes sont détaillées au chapitre 1, qui constitue un état de l'art. Certaines formes de coopération entre les deux paradigmes y sont en effet recensées et classifiées selon trois méthodologies distinctes : les méthodes *exactes incomplètes*, les approches travaillant sur des *configurations partielles*, les méthodes de *grands voisinages*. Nous nous sommes ainsi avant tout intéressés aux approches combinant résolution exacte et approchée au sein d'un processus global. Les deux approches complémentaires coopèrent ici directement dans le sens où les méthodes exposées intègrent au sein d'un processus basé sur l'une des deux méthodologies une(des) composante(s) appartenant à l'autre méthodologie.

A la suite de ce chapitre, nous poursuivons dans un premier temps par la présentation d'une méthode de grands voisinages (3*e* catégorie), baptisée LSSPER (Local Search (with) Sub-Problem Exact Resolution), et son application à deux problèmes : le problème d'*ordonnancement de projet sous contraintes de ressources*, qui constitue un problème académique classique et abondamment étudié, et un problème particulier d'*affectation de fréquences* en réseau hertzien, qui introduit une nouvelle problématique, basée sur des considérations de terrain, proposée par le CELAR[1]. Notre objectif consiste à démontrer la validité de l'approche proposée non seulement en terme de problèmes académiques mais également réels.

Dans un cadre général, LSSPER est basée sur la génération puis la résolution, à l'aide d'une procédure complète, de sous-problèmes successifs du problème global. La méthode consiste donc bien en une approche coopérative, la composante exacte étant ici intégrée au sein d'un processus de recherche locale, dont nous étudierons les performances sur les deux problèmes sus-nommés. Nous examinerons ainsi au cours des chapitres 2 et 3 les spécifications de la méthode nécessaires à la prise en compte des caractéristiques de ces deux problèmes bien distincts. Nous verrons ainsi que, si le schéma général de LSSPER est relativement bien adapté à la résolution du problème d'ordonnancement, en dépit d'une certaine lenteur de résolution, l'obtention d'une procédure efficace sur le problème d'affectation de fréquences nécessite de nombreuses adaptations. Celles-ci sont notamment rendues obligatoires par la grande taille des instances considérées ainsi que par la difficulté de prise en compte de certaines contraintes.

Le dernier chapitre est finalement dévolu à la présentation d'une méthode de résolution qui constitue le pendant de cette première approche (1ère et 2e catégories). Celle-ci se propose en effet d'intégrer au sein d'un processus arborescent exhaustif des composants heuristiques dans le but, principalement, de restreindre la taille de l'espace de recherche exploré. Cette approche est basée sur l'emploi de la procédure de *Resolution Search* [Chvátal 1997], qui constitue une alternative aux recherches arborescentes classiques. Nous présenterons ainsi dans un premier temps les particularités de cette méthodologie, puis son application en tant que méthode exacte associée à des techniques de propagation de contraintes au problème particulier de *coloration* de graphes des reines. Nous poursuivrons ensuite par la présentation d'un schéma incomplet s'appuyant sur la considération de caractéristiques de symétrie bien particulières des instances de ce problème. Nous terminerons par la présentation de perspectives d'amélioration de la méthode et de poursuite des travaux d'intégration plus poussée de composants heuristiques dans Resolution Search.

[1] Centre d'Electronique de l'ARmement

# Chapitre 1

# Coopération entre recherche exacte et approchée pour la résolution de problèmes d'optimisation combinatoire

---

*L'espoir fait vivre... mais l'attente fait mourir.*
*Voici un proverbe qui prend ici tout son sens. En effet, quoi de plus désespérant que d'attendre indéfiniment la solution optimale d'un problème combinatoire ? Nous prendrons donc le parti au cours de ce chapitre de préférer l'espoir en présentant, notamment, une méthode de grand voisinage au nom aussi évocateur que possible, LSSPER. L'idée principale de cette dernière est de concilier des principes de résolution exacte et heuristique. Ce chapitre se veut ainsi être un état de l'art présentant les diverses formes de coopération possible entre ces deux paradigmes.*

---

## 1.1 Problématique

Les problèmes combinatoires NP-difficiles (ou NP-complets) de taille importante sont difficiles à appréhender par une méthode exacte complète. C'est pourquoi de nombreuses approches délaissent ce schéma pour se tourner vers une résolution incomplète de ces problèmes dans l'optique d'obtenir des solutions, pour lesquelles la garantie d'optimalité n'est certes plus assurée, mais dans des temps de calcul "raisonnables". Ainsi le champ des méthodes heuristiques a-t-il connu un essor formidable au cours des dernières décennies. Celles-ci sont d'une grande diversité : parmi elles on retrouve notamment les algorithmes de recherche locale (recherche tabou, recuit simulé,...) qui travaillent sur le voisinage d'une solution, les approches évolutionnistes qui considèrent des populations de solutions, etc. Cependant, les méthodes exactes demeurent performantes dès lors qu'il s'agit de résoudre des problèmes de taille moindre ou présentant des caractéristiques bien définies.

Il semble alors naturel de se tourner vers des méthodes "hybrides", faisant coopérer des principes issus de ces deux approches complémentaires dans l'optique de tirer parti des avantages que chacune d'elles peut apporter : d'une part, le caractère systématique de la composante exacte, en plus de la preuve d'optimalité qu'elle apporte, d'un autre côté, la rapidité d'exécution de la composante heuristique, ainsi que son aspect éventuellement non déterministe. On peut globalement distinguer deux façons distinctes de procéder à cette hybridation.

Celle-ci peut ainsi revêtir la forme d'une coopération séquentielle : les deux processus s'enchaînent alors l'un après l'autre, le deuxième tirant parti des informations procurées par le premier. Le séquencement peut avoir lieu dans les deux sens mais le cas où la première phase consiste en un processus heuristique est une forme de coopération très peu répandue, si l'on exclut le simple calcul d'une solution réalisable par une heuristique avant l'exécution d'une procédure de séparation et évaluation. Nous pouvons toutefois citer la méthode de Crawford [Crawford 1993], appliquée à la résolution de problèmes de satisfaction de contraintes. Un processus de recherche locale est préalablement exécuté afin de déterminer des poids associés aux clauses du problème. Ces informations sont ensuite intégrées dans le mécanisme de branchement de la méthode exacte afin de favoriser les variables associées aux clauses de poids élevé. Cette coopération permet de réduire la taille de l'espace de recherche de manière non négligeable. Le séquencement inverse consiste quant à lui à améliorer les résultats obtenus par une approche exacte (incomplète) à l'aide d'une méthode heuristique. Ainsi, la recherche arborescente est ici simplement utilisée pour générer des solutions initiales sur lesquelles une phase d'amélioration heuristique est alors appliquée. Nous ne nous étendrons cependant pas sur le sujet et choisirons de nous focaliser sur l'autre forme d'hybridation qui consiste en une coopération imbriquée des deux procédés : ceux-ci sont alors intégrés de concert au sein d'une méthode générale.

L'enjeu premier sous-jacent à l'intégration d'une composante heuristique au sein d'un processus exhaustif concerne la réduction de la taille de l'espace de recherche à explorer. Les techniques classiques de réduction, si elles assurent de ne pas exclure une région susceptible de contenir la solution optimale, sont en effet bien souvent impuissantes à rendre l'espace de recherche résiduel suffisamment petit pour permettre son exploration complète en des temps raisonnables. Il est alors nécessaire d'intégrer au sein du processus une composante décisionnelle qui va permettre d'infléchir la recherche vers certaines régions de l'espace, à l'exclusion de toutes les autres. Il s'agit donc de

déterminer des critères, heuristiques "à priori", et/ou d'apprentissage "à posteriori", permettant d'orienter la recherche de façon adéquate vers les régions les plus à même de contenir des solutions de bonne qualité. Nous développerons ce point en section 1.2.

Un autre enjeu consiste à atténuer les inconvénients liés à la systématicité des méthodes exactes par l'introduction de composants heuristiques. Il s'agit alors de définir un schéma hybride de déplacement dans l'espace de recherche qui concilie des mouvements issus de la recherche systématique et de la recherche locale. Ces méthodes, qui présentent la particularité de travailler sur des configurations partielles, seront exposées en section 1.3.

Finalement, un dernier enjeu consiste à utiliser une méthode exacte au sein d'un processus heuristique pour explorer de façon approfondie des voisinages de taille importantes. Nous clôturerons ainsi cet état de l'art par la présentation de la méthode LSSPER en section 1.4. Celle-ci se veut être un schéma d'unification des approches de recherche de grands voisinages. L'idée principale de la méthode repose sur la génération, puis la résolution à l'aide d'une méthode exacte, de sous-problèmes successifs du problème initial. Nous verrons ainsi que le point crucial de cette approche réside dans la définition appropriée des sous-problèmes, ainsi que dans la mise en place de stratégies de guidage de la recherche adaptées.

## 1.2 Les méthodes exactes incomplètes

### 1.2.1 Introduction

Les méthodes de résolution exactes reposent sur l'énumération implicite de l'ensemble des solutions du problème à résoudre, énumération qui prend le plus communément la forme d'une recherche arborescente. C'est par exemple le cas de la procédure de séparation et évaluation en programmation mathématique et de la méthode de backtracking en programmation par contraintes. L'espace de recherche est ainsi exploré de façon bien déterminée par un parcours dans un arbre. La manière dont le parcours est réalisé dépend de la stratégie générale d'exploration : recherche en profondeur d'abord, en largeur d'abord... Une fois la stratégie fixée, il s'agit alors à chaque nœud de décider quelle sera la prochaine région de l'espace à explorer, ce qui est souvent réalisé par le choix d'une variable de branchement et de sa valeur d'affectation (règle de branchement ou de séparation). Divers critères peuvent être retenus pour la prise de décision, et de ces derniers dépend bien souvent l'efficacité de la procédure. Parmi tous les critères existants, nous nous intéresserons à ce que l'on appelle communément, en programmation par contraintes, les heuristiques de guidage de la recherche : à chaque nœud de l'arbre, des considérations d'ordre heuristique permettent de désigner, à priori, la variable de branchement, ainsi que sa valeur d'affectation, permettant de conduire aux solutions les plus prometteuses. Il en ressort que du choix de l'heuristique et de son adéquation au problème considéré vont découler les performances de la méthode. Ce point est d'autant plus critique lorsque l'exploration de l'arbre de recherche devient nécessairement partielle, en regard des temps d'exécution requis. De toute évidence, il s'agit alors de se diriger d'abord vers les solutions de meilleure qualité.

Lorsque l'heuristique s'avère très performante, il devient possible de n'effectuer qu'une recherche très parcellaire autour de la solution que celle-ci procure, voire, dans le cas le plus extrême, de s'en

contenter. C'est le cas par exemple de la méthode de recherche arborescente incomplète la plus primaire, consistant en une unique descente dans l'arbre de recherche qui suit à chaque nœud le choix de l'heuristique. Nous appellerons cet algorithme 1-samp [Harvey 1994], [Smith 1993]. Lorsqu'on recherche une solution réalisable pour des problèmes dont la densité de solutions est élevée, un tel algorithme conduira quasi-systématiquement au succès. Cependant, lorsque la densité de solutions diminue, ou lorsqu'on cherche une solution optimisant un critère donné, il devient nécessaire d'employer d'autres méthodes permettant d'explorer une portion plus importante de l'espace de recherche.

Les méthodes d'échantillonnage consistent à suivre des chemins aléatoires dans l'arbre de recherche à partir de la racine jusqu'à ce qu'une solution soit rencontrée ou le nœud courant écarté (détection d'une inconsistance ou conditions sur les bornes) : échantillonnage aléatoire (iterative sampling ou ISAMP) [Langley 1992]. L'application d'une telle procédure à des problèmes de job-shop [Crawford 1994] a montré une certaine efficacité. De façon générale cette procédure semble exhiber de bons résultats lorsque la densité des solutions est élevée et que ces dernières sont réparties de façon à peu près uniforme dans l'arbre de recherche. La composante aléatoire permet en effet de parcourir des portions potentiellement très diverses de l'arbre et augmente de fait la probabilité de rencontrer une solution. Cependant, l'indépendance de la méthode vis-à-vis de toute considération heuristique pose les questions de son applicabilité à des problèmes pour lesquels la densité de solutions est faible.

Une parade consiste alors à diminuer, voire annihiler, l'importance de la composante aléatoire. L'idée générale est alors d'explorer le voisinage de la décision réalisée par l'heuristique à chaque étape du processus. Un tel résultat peut être obtenu par divers moyens : en biaisant le choix préconisé par l'heuristique à l'aide d'une composante aléatoire (random biased sampling) [Kolisch 1996], en réalisant une évaluation lookahead de la qualité des branches (cf. section1.2.2) ou encore en considérant un sous-ensemble de branches proches de la branche empruntée par l'heuristique (cf. sections 1.2.3, 1.2.4 et 1.2.5).

### 1.2.2 Algorithmes de lookahead

Les heuristiques permettant d'évaluer l'intérêt d'une branche sont souvent myopes, en cela que les choix qu'elles préconisent reposent uniquement sur les conséquences immédiates qu'ils amènent. Les algorithmes de lookahead tentent de remédier à cette déficience en s'inspirant de la théorie des jeux : un joueur sélectionne son prochain mouvement en se projetant $k$ mouvements plus loin et en choisissant celui pour lequel la pire situation atteignable est la meilleure. De la même façon, avant d'évaluer une branche, un algorithme de lookahead va tenter de descendre plus ou moins profondément dans celle-ci afin d'en retirer des informations supplémentaires qui vont lui permettre d'affiner son évaluation.

Le schéma $H_k$, introduit par Sourd [Sourd 2001], se place parmi ces méthodes. Celui-ci se base sur l'utilisation incrémentale de l'heuristique de guidage de la recherche $\mathcal{H}$. De façon plus précise, si l'on considère le schéma $H_k$, les fils d'un nœud décisionnel de l'arbre de recherche sont évalués par l'intermédiaire du schéma $H_{k-1}$, dont les fils respectifs sont eux-mêmes évalués d'après le schéma $H_{k-2}$, etc. La descente est alors réalisée vers le fils qui comporte la meilleure évaluation et le

schéma $H_k$ est réappliqué à partir de ce nœud. L'imbrication des appels s'achève au schéma $H_0$ qui correspond à l'évaluation directe par l'heuristique $\mathcal{H}$. Un exemple de schéma $H_1$ est donné en Figure 1.1 sur un arbre binaire de profondeur 4.

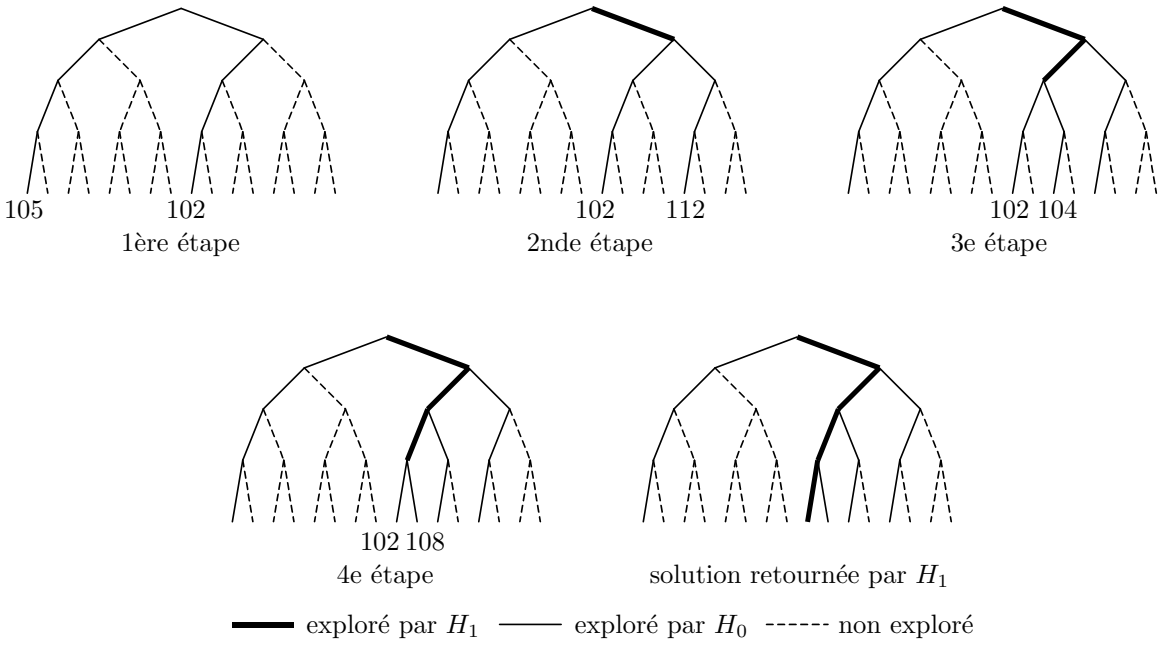


FIG. 1.1 – *$H_1$ sur un arbre binaire de profondeur 4*

### 1.2.3 Recherche à divergences limitées et méthodes connexes

Une divergence est un point de décision dans un arbre de recherche pour lequel le choix réalisé est différent de celui préconisé par l'heuristique. Pour des raisons pratiques, nous considérerons dans la suite de notre propos que la descente vers le fils situé à l'extrême gauche d'un nœud est conforme au choix de l'heuristique. Par analogie, toute autre descente, c.-à-d. vers un fils droit, constitue une divergence.

La paternité de la méthode originelle de recherche à divergences limitées (ou LDS pour Limited Discrepancy Search) revient à Harvey et Ginsberg [Harvey 1995b]. Celle-ci s'adresse aux cas d'échecs de 1-samp. Ces derniers peuvent être imputés à la prise d'un certain nombre de mauvaises décisions. Cependant, les conséquences de la prise d'un nombre restreint de mauvaises décisions peuvent être éliminées par une recherche systématique de tous les chemins de l'arbre qui diffèrent du chemin emprunté par l'heuristique en au plus un petit nombre de points de décision : les divergences. Ainsi la procédure explore-t-elle de façon itérative tous les chemins de l'arbre de recherche par nombre croissant de divergences qu'ils comportent. Plus précisément, à l'itération $s$, LDS explore tous les chemins comportant au plus $s$ divergences par rapport au chemin emprunté par l'heuristique. De la valeur de $s$ dépend bien évidemment la portion de l'arbre de recherche ex-

ploré : de l'itération 0, qui reconstitue le chemin emprunté par l'heuristique, à l'itération $n$, qui correspond à un parcours exhaustif de l'arbre, le champ des possibilités est donc vaste.

De fait, LDS procure une grande souplesse d'utilisation. En effet, un paramétrage adéquat permet de mettre l'accent sur la caractéristique désirée : rapidité d'exécution, au détriment éventuel de la qualité des solutions, lorsque peu d'itérations sont réalisées ou obtention de solutions de meilleure qualité, associées à des temps d'exécution plus élevés, lorsque ce nombre augmente. Cette souplesse, alliée à la simplicité du schéma général, lui permet d'être intégrée de façon intuitive au sein de processus plus complexes. Elle peut ainsi être couplée avec des techniques permettant d'améliorer son efficacité, comme Bounded Backtrack Search (BBS) [Harvey 1995a] (principe). Ses performances peuvent également être renforcées en relançant son exécution pour des ordres d'instantiation des variables différents ou en faisant varier l'heuristique de branchement choisie. Elle peut également être utilisée au sein d'algorithmes de recherche locale pour explorer le voisinage d'une solution donnée, ainsi que nous pourrons le constater au cours de sections ultérieures.

Cependant, cette technique comporte l'inconvénient de revisiter à chaque itération des nœuds terminaux déjà explorés au cours des itérations précédentes. Une variante de LDS permet de gommer ce défaut : c'est la procédure ILDS (Improved Limited Discrepancy Search) mise au point par Korf [Korf96]. Celle-ci ne génère à chaque itération que les chemins comportant exactement $s$ divergences en branchant systématiquement à droite lorsque la profondeur restante de l'arbre à explorer est inférieure ou égale au nombre restant de divergences à accomplir. Cette amélioration permet de réduire le nombre de fois où les nœuds intérieurs sont revisités par l'algorithme, les feuilles n'étant ici visitées qu'une seule fois, contrairement à la version originale. Un exemple d'exécution de la procédure conduisant au parcours exhaustif d'un arbre binaire de profondeur 4 est donné en Figure 1.2.

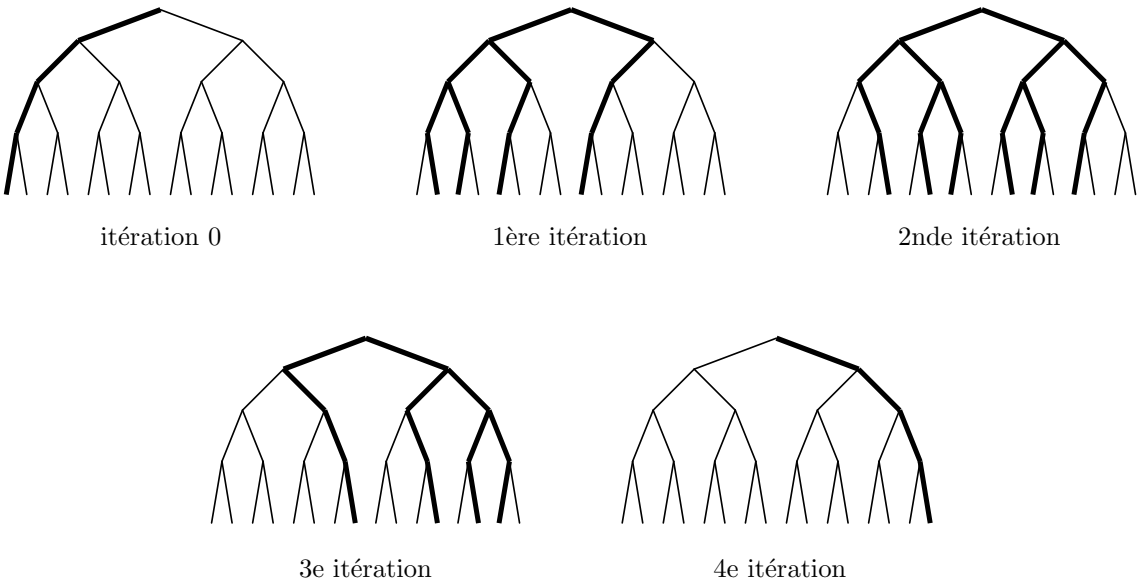


FIG. 1.2 – *ILDS sur un arbre binaire de profondeur 4*

Comme on peut le constater clairement sur cet exemple, LDS (ou ILDS) possède un autre avantage en cela qu'elle permet de parer aux inconvénients liés aux procédures classiques de recherche en profondeur d'abord. En effet, les auteurs supposent que les mauvais choix réalisés par l'heuristique sont plus à même d'apparaître en haut de l'arbre car celle-ci possède alors moins de connaissances. Le backtrack chronologique, couramment invoqué dans les procédures classiques, est donc potentiellement amené à perdre énormément de temps à explorer un sous-arbre qui ne peut pas contenir de solutions, l'erreur étant commise à un niveau supérieur de l'arbre de recherche (phénomène de trashing [Hogg 1996]). LDS permet donc de combler cette lacune en traitant toutes les divergences de la même façon, indépendamment de la profondeur à laquelle elles apparaissent.

Une autre variante de LDS se propose d'accentuer encore davantage la recherche de divergences situées à un niveau élevé de l'arbre de recherche. Cette procédure de recherche à divergences de profondeur bornée (Depth-bounded Discrepancy Search), revenant à Walsh [Walsh 1997], se veut même l'antithèse de la recherche en profondeur d'abord. Celle-ci combine LDS et "iterative deepening search" [Korf 1985] et favorise la recherche de divergences en haut de l'arbre par le biais d'une "borne de profondeur" qui est incrémentée à chaque itération de la méthode. Les divergences se situant à une profondeur supérieure à la borne sont interdites. Plus précisément, à l'itération $s+1$, DDS explore les chemins de l'arbre pour lesquels les divergences apparaissent à la profondeur $s$ ou inférieure. De même que ILDS les nœuds terminaux ne sont visités qu'une seule fois : à la profondeur $s$, on branche à droite ; au-dessous, on choisit systématiquement la branche gauche, au-dessus toutes les branches peuvent être empruntées indifféremment. Un exemple d'exécution est donné en Figure 1.3.

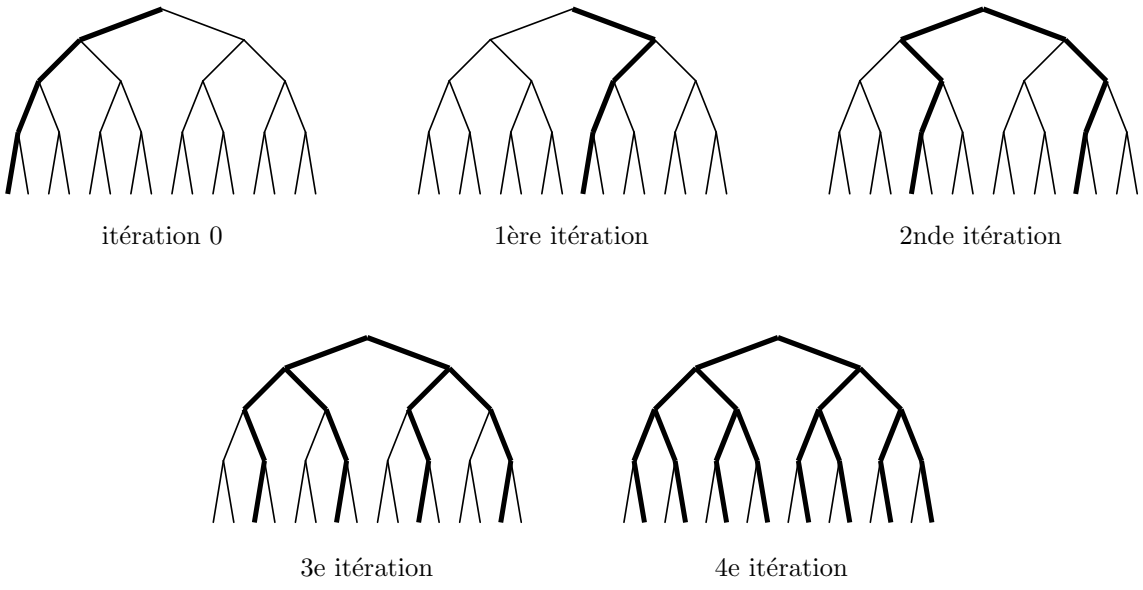


FIG. 1.3 – *DDS sur un arbre binaire de profondeur 4*

La spécialisation de DDS, si elle permet de corriger à coût négligeable les erreurs précoces,

est néanmoins très pénalisante lorsqu'il s'agit de surmonter une erreur réalisée profondément dans l'arbre de recherche. Une possibilité de palier à ce désagrément réside dans son hybridation avec BBS. Cette dernière procédure permet en effet de récupérer rapidement des erreurs réalisées profondément dans l'arbre pour un coût additionnel modique [Harvey 1995a]. Cette hybridation apporte ainsi une double compétence, DDS prenant en charge les erreurs commises en haut de l'arbre tandis que BBS permet de gérer celles commises en bas de l'arbre.

Une autre variante de LDS, nommée recherche à crédit limité (Credit Search [Beldiceanu 1999]), se propose de pénaliser davantage l'emprunt de branches éloignées de celle empruntée par l'heuristique. Ainsi, suivant le formalisme que nous avons décidé d'utiliser au cours de ce chapitre, les fils d'un nœud sont classés, par évaluation décroissante, de la gauche vers la droite. Ce classement permet d'assigner à chaque divergence un coût qui est égal au rang de la branche considérée, le rang 0 étant affecté à la branche la plus à gauche. La recherche consiste alors à parcourir l'ensemble des chemins pour lesquels le coût total induit par les divergences n'excède pas la limite fixée par le crédit.

Cette approche permet de combler de façon élégante le flou laissé par les approches précédemment évoquées quant au traitement des divergences lorsque l'arbre de recherche n'est pas binaire. En effet, dans le cas de variables binaires, la définition d'une divergence est sans équivoque car il n'y a alors pour une variable donnée que deux possibilités d'affectation. Cependant, lorsque le domaine des variables augmente, celle-ci devient sujette à l'interprétation qui en est faite. Il s'agit alors de définir si une divergence revient uniquement à considérer le choix suivant le plus attractif, ou bien l'ensemble des choix possibles, ou encore si elle se situe entre ces deux extrêmes. Un compromis peut alors consister à considérer une liste de candidats restreints.

### 1.2.4 Liste de candidats restreints

À chaque point de décision de l'arbre de recherche, l'heuristique de guidage désigne une branche de prédilection parmi toutes les possibilités qui lui sont offertes. Elle permet également d'indiquer la qualité respective de chacune des autres branches. Ainsi, dans le cas d'arbres binaires, les deux branches peuvent ainsi être quasi-équivalentes ou bien l'une consister en une bonne option tandis que l'autre en une très mauvaise. Dans le cas de branchements plus larges (non binaires), l'heuristique pourraient ainsi considérer que certaines branches sont des candidats sérieux, tandis que toutes les autres seraient irrémédiablement rejetées. L'idée de la liste de candidats restreints (Restricted Candidate List) consiste à limiter la recherche aux branches proches du choix heuristique. Plus précisément, toute branche qui ne s'éloigne pas de la décision préconisée par l'heuristique d'un certain ratio $\alpha$ est sélectionnée parmi les candidats. Le paramétrage de la valeur de $\alpha$ permet ainsi de générer un sous-arbre plus ou moins restreint sous le nœud courant : du sous-arbre correspondant au chemin emprunté par l'heuristique lorsque $\alpha$ vaut 0 à la totalité de l'arbre de recherche quand à l'opposé $\alpha$ vaut 1. Pour des valeurs intermédiaires, le sous-arbre contient des chemins qui seront plus ou moins proches du chemin heuristique.

De tels sous-arbres peuvent être explorés de façon systématique ou non. Parmi les méthodes non systématiques, la procédure GRASP (Greedy Randomised Adaptative Search Procedure) [Feo 1995] utilise une composante aléatoire pour parcourir le sous-arbre défini par la RCL. À

chaque nœud décisionnel, une branche est sélectionnée de façon aléatoire parmi les branches candidates conformément à une distribution probabiliste qui attribue des probabilités décroissantes à mesure que la branche s'éloigne de celle préconisée par l'heuristique. Les branches proches du choix de l'heuristique sont de fait plus susceptibles d'être empruntées que celles qui en diffèrent grandement. Une fois une solution complète atteinte, celle-ci est post-optimisée par application d'une méthode de descente puis le processus est réitéré à partir de la racine jusqu'à ce qu'un critère d'arrêt soit vérifié.

La méthode peut être spécifiée précisément par le biais de la valeur de $\alpha$ et de la distribution probabiliste choisie. Des études [Prais 1998] montrent par exemple qu'il est plus efficace de considérer une valeur variable de $\alpha$, en démarrant le processus avec une valeur proche de 0 puis augmentant cette dernière au fur et à mesure des itérations afin de permettre l'acceptation de choix localement mauvais. Une autre possibilité consiste à faire varier la valeur de $\alpha$ en fonction de la profondeur du nœud courant : celle-ci peut ainsi être augmentée à mesure que la recherche s'enfonce profondément dans l'arbre de recherche.

Des approches systématiques peuvent également être utilisées pour parcourir une portion plus ou moins importante des branches de la RCL. C'est le cas de la méthode de beam search.

### 1.2.5 Beam search

L'approche de beam search (BS) est une heuristique bien établie qui est issue du monde de l'intelligence artificielle. Les premières tentatives d'application à l'optimisation combinatoire sont relativement anciennes [Ow 1988] et [Ow 1989]. La méthode repose sur une recherche arborescente tronquée couplée à une stratégie de type en largeur d'abord où seuls les $w$ nœuds les plus prometteurs à une profondeur donnée sont effectivement explorés : $w$ est appelé largeur de faisceau. L'évaluation des nœuds est généralement réalisée en deux temps : une première phase de filtrage permet de sélectionner à un coût modeste un certain nombre de nœuds qui sont ensuite évalués de façon plus fine et $w$ d'entre eux sont conservés pour la suite de l'exploration de l'arbre. Figure 1.4 représente les sous-arbres potentiellement explorés par BS pour une valeur de faisceau quelconque.

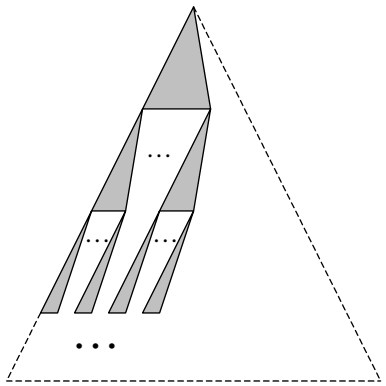

FIG. 1.4 – *BS sur un arbre quelconque*

L'inconvénient majeur de BS, comme des autres approches présentées jusqu'alors, est qu'une erreur dans le processus d'évaluation, conduisant à la fermeture d'un nœud menant à une solution optimale ou proche de l'optimum, est irréversible. La solution retournée par le processus global peut ainsi être fortement éloignée de la solution optimale dans les cas les moins favorable. Afin d'éviter l'apparition d'un tel cas de figure, il convient alors de choisir une valeur de largeur de faisceau suffisamment élevée, ce qui ralentit considérablement le processus. Une façon de pallier ce désagrément a été proposée par Della Croce et al. [Della Croce 2004] : il s'agit d'insérer dans le processus global une étape de récupération (Recovering Beam Search) qui va rechercher, à chaque niveau de l'arbre de recherche, des solutions partielles dominantes par rapport à celles sélectionnées par le faisceau. Cette phase de récupération est réalisée par l'application d'opérateurs d'échange aux solutions partielles examinées par le faisceau et la confrontation des diverses solutions ainsi obtenues entre elles. Les solutions retenues pour la poursuite de la recherche dépendront des conditions de dominance, forcément dépendantes du problèmes, définies au préalable.

La méthodologie employée consiste à rechercher à chaque niveau de l'arbre des solutions voisines, respectivement à un opérateur d'échange, de solutions examinées par le faisceau : le processus peut ainsi être entrevu comme une alternance de phases de descente et de phases de recherche locale dans le sous-espace des configurations partielles de taille donnée. Il permet, dans une certaine mesure, de pallier les inconvénients d'une recherche systématique. D'autres approches tentent de renforcer cet aspect en permettant une diversité de mouvements plus grande encore à chaque nœud de l'arbre de recherche : Lhomme [Lhomme 2004] les classifie sous la dénomination de méthodes de recherche non-systématiques sur des configurations partielles.

## 1.3 Recherche locale sur des configurations partielles

### 1.3.1 Introduction

Afin de pallier les erreurs, éventuellement désastreuses (trashing), liées à la systématicité des approches arborescentes classiques, une panoplie de méthodes a vu le jour, qui abandonnent cet aspect-là. Ces algorithmes, non-systématiques donc, travaillent sur des affectations partielles des variables. Ils sont capables de sauter d'un nœud à l'autre de l'arbre de recherche par des mécanismes qui diffèrent des mouvements classiques de descente et remontée. En fait, ces derniers ne sont pas sans rappeler ceux invoqués au cours de méthodes de recherche locale : ponctuellement, des mouvements de voisinage vont permettre d'améliorer ou réparer une solution partielle. Le fait que ces algorithmes travaillent sur des configurations incomplètes offre en outre la possibilité, contrairement aux approches de recherche locale, de mettre en place des techniques permettant de restreindre les choix possibles pour les variables non encore assignées et, de fait, de réduire la complexité du voisinage considéré.

La figure 1.5 présente de façon simplifiée le principe de fonctionnement des diverses approches pouvant être rencontrées. Les deux premières méthodes (parties gauche et centrale de la figure) réalisent l'exploration de l'arbre par combinaison de deux types de mouvements : recherche arborescente pour la descente dans l'arbre et recherche locale pour l'amélioration ou la réparation de solutions partielles. La première technique est utilisée lors de la résolution de problèmes d'optimi-

sation tandis que le deuxième procédé, typique des problèmes de satisfaction, consiste à réparer (c.-à-d. rendre à nouveau consistante) une solution partielle irréalisable vis-à-vis des contraintes du problème. Le troisième schéma (partie droite de la figure) consiste en une approche plus singulière. Celle-ci ne fait en effet intervenir que des mouvements de type recherche locale pour le parcours de l'arbre.

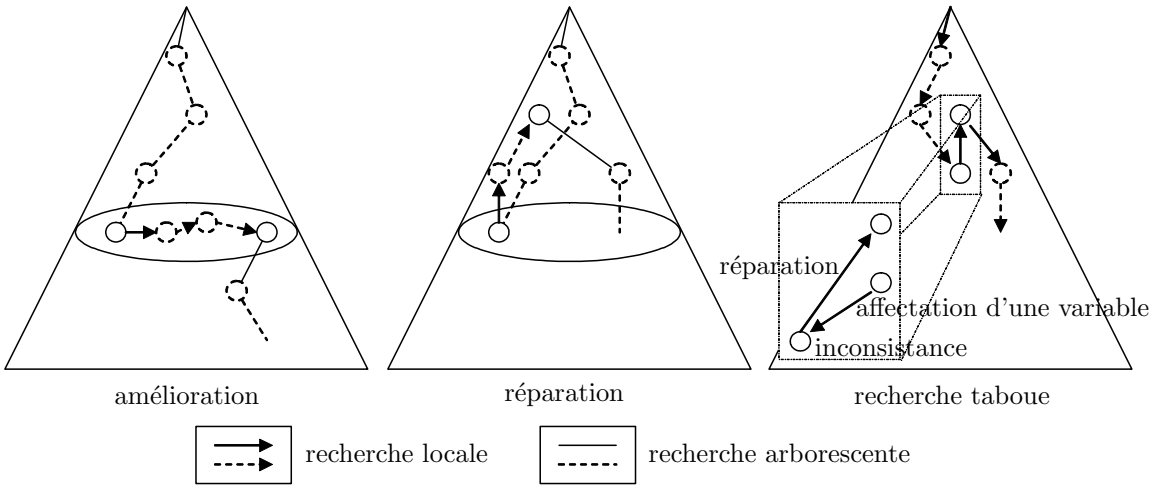


FIG. 1.5 – *Recherche locale au sein d'un arbre de recherche*

Nous détaillons quelques une de ces méthodes dans les sections suivantes. En section 1.3.2, nous présentons brièvement des approches qui utilisent la recherche locale comme technique d'amélioration de solutions partielles. Nous développons ensuite plus en détail diverses approches pour lesquelles l'outil de recherche locale sert à la réparation de solutions partielles en section 1.3.3. Nous terminons finalement en section 1.3.4 par la présentation de l'approche particulière de recherche taboue à voisinage consistant qui fait exclusivement intervenir des mouvements de voisinage.

### 1.3.2 La recherche locale comme outil d'amélioration de solutions partielles

Russell [Russell 1995] propose une méthode pour résoudre des problèmes de tournées de véhicules. Celle-ci consiste à réaliser une série de mouvements de type recherche locale toutes les $t$ étapes d'insertion du processus global : chaque mouvement tente d'améliorer un plan partiel par une autre. Le voisinage est restreint aux plans partiels qui visitent les mêmes clients, mais dans un ordre différent, que le plan partiel considéré.

Une autre application dans le domaine de problèmes de tournées est due à Caseau et Laburthe [Caseau 1999b]. Leur méthode, appelée ILO (Incremental Local Optimization), qui applique une phase de recherche locale après chaque insertion, est comparée à une méthode consistant à améliorer la solution obtenue par un algorithme glouton à l'aide de la recherche locale. Ils montrent que ILO est supérieure tant au niveau des temps d'exécution que de la qualité des solutions produites.

### 1.3.3 La recherche locale comme outil de réparation de solutions partielles

Prestwich [Prestwich 2004] présente une méthode pour la résolution de problèmes de satisfaction de contraintes qu'il qualifie de recherche locale dans le sous-espace des configurations partielles consistantes. Il appelle celle-ci Incomplete Dynamic Backtracking (IDB) en référence à la méthode de Dynamic Backtracking (DB) [Ginsberg 1993] dont elle s'inspire.

Ainsi que cette dernière, IDB permet en effet de réorganiser de façon dynamique l'arbre de recherche mais, à son inverse, ne conserve pas l'aspect de complétude. Ainsi, DB, algorithme de backtracking dit intelligent, permet, au cours d'une phase de remontée, de désinstancier une variable assignée à un nœud supérieur de l'arbre de recherche sans toucher aux affectations réalisées après celle-ci. Cependant, la conservation de la complétude restreint le choix de ladite variable. Une version améliorée (Partial Order Dynamic Backtracking [Ginsberg 1994]) permet d'augmenter la flexibilité dans le choix de la variable mais impose tout de même des restrictions pour assurer la complétude, à défaut d'utiliser une quantité de mémoire exponentielle qui elle seule permettrait la liberté totale. À l'inverse, IDB ne pose aucune contrainte sur la variable qui sera désinstanciée et ne retient pas non plus d'informations quant aux régions de l'espace de recherche déjà explorées, abandonnant ainsi l'aspect de complétude au profit de plus de souplesse. Au terme d'une phase de descente, c.-a-d. lorsqu'il n'est plus possible de descendre dans l'arbre de recherche sans provoquer d'inconsistance, IDB désinstancie un nombre $b \geq 0$ de variables, puis recommence la descente à partir de la nouvelle solution partielle ainsi obtenue.

Une particularité d'IDB est qu'il est couplé à une technique de forward-checking[1] qui restreint les affectations possibles pour la variable de branchement aux seules valeurs de son domaine n'induisant pas d'inconsistance à la suite de l'étape de propagation. IDB travaille ainsi exclusivement dans l'espace des configurations partielles consistantes, les mouvements réalisés au sein de cet espace étant dictés par la valeur de $b$ (appelé paramètre de bruit) et les diverses heuristiques utilisées : choix de la variable de (dé)branchement, choix de la valeur d'affectation.

À l'inverse, d'autres approches travaillent dans l'espace défini par l'ensemble des configurations partielles, consistantes ou non. Ainsi de la méthodologie proposée par Schaerf [Schaerf 1997] qui combine les techniques de backtracking et de recherche locale : chaque fois qu'une inconsistance est rencontrée à l'issue d'une phase de descente classique dans l'arbre de recherche, un processus de recherche locale est appliqué à la configuration partielle courante afin de réparer celle-ci. L'exploration du sous-espace des configurations partielles est guidée par une fonction de coût basée sur divers composants : la distance à la réalisabilité de la solution partielle, un facteur de look-ahead, et, si besoin, une borne inférieure de la fonction objectif, ce qui permet d'adapter la méthode à la résolution de problèmes d'optimisation.

Une approche, à l'initiative de Jussien et Lhomme [Jussien 1999], appelée path-repair, propose d'enrichir ce schéma à la façon d'IDB. Les modifications apportées concernent l'ajout de techniques de propagation de contraintes, ce qui restreint l'examination des seules configurations partielles consistantes, ainsi que le stockage de nogoods ((sous-)instanciations des variables inconsis-

[1] élimination des valeurs inconsistantes des domaines des variables non encore instanciées par le biais des contraintes les liant à la dernière variable instanciée

tantes avec les contraintes du problème) qui permettent, le cas échéant, de réaliser une exploration complète de l'espace de recherche. Une version taboue de cet algorithme permet de restreindre le voisinage considéré, toute solution constituant une extension d'un des nogoods stockés dans la liste taboue en étant écartée.

Nous étudierons au cours du chapitre 4 une méthode exhaustive travaillant sur des configurations partielles, Resolution Search [Chvátal 1997]. A l'instar des méthodes que nous venons de présenter, celle-ci fait intervenir des mouvements classiques de descente dans l'arbre de recherche mais la reprise de la recherche suite à la découverte d'un échec est ici assurée par la réalisation d'un mouvement défini par des considérations vis-à-vis d'une famille de nogoods.

### 1.3.4 Recherche taboue à voisinage consistant

Dupont et al. [Dupont 2004] proposent une méthode taboue qui travaille dans l'espace des configurations partielles consistantes. Le voisinage considéré à chaque itération comporte toutes les solutions atteignables à partir de la solution courante en appliquant un double opérateur : instanciation d'une variable non encore affectée à une valeur non taboue de son domaine puis réparation éventuelle de la solution afin de maintenir la consistance (les variables en conflit avec la nouvelle affectation sont désinstanciées). Une originalité de la méthode concerne l'évaluation de la qualité d'une solution. La fonction objectif utilisée concerne en effet la maximisation du nombre de variables instanciées : le voisin vers lequel sera réalisé le mouvement (le meilleur) est donc celui qui implique, pour des raisons de préservation de la consistance, le moins de désinstanciations ultérieures de variables déjà affectées. La liste taboue, gérée de façon dynamique, est maintenue à l'issue de chaque mouvement par l'ajout des valeurs des domaines des variables non instanciées qui entreraient en conflit avec la dernière affectation réalisée. Ainsi, le dernier mouvement réalisé est préservé durant un certain nombre d'itérations. Un critère d'aspiration permet de lever le statut tabou d'un mouvement permettant d'améliorer la valeur de la fonction objectif.

Cette méthode se démarque quelque peu de celles présentées précédemment en cela qu'elle ne fait jamais appel à des mouvements de type recherche arborescente. L'aspect hybride est ici apporté par le travail sur les configurations partielles et les techniques de filtrage qu'il permet de mettre en place. Le voisinage est ainsi restreint à l'aide d'outils exacts. Une autre forme d'hybridation, à laquelle nous allons consacrer la fin de ce chapitre, utilise également des techniques issues de la recherche systématique, non pas pour définir le voisinage, mais pour le parcourir. Ces méthodes définissent à chaque itération un sous-problème du problème initial et le résolvent à l'aide d'une méthode exacte, éventuellement incomplète. Nous tenterons au cours de la section suivante de définir une procédure d'unification de ce type d'approches : LSSPER.

## 1.4 LSSPER : une tentative d'unification des méthodes de recherche de grands voisinages

### 1.4.1 Introduction

L'efficacité des méthodes complètes sur des problèmes de taille modeste ou présentant des con-

traintes bien particulières n'est plus à démontrer. De nouvelles techniques, ainsi qu'un accroissement de la puissance de calcul, permettent continuellement de repousser les limites de ces approches. Dès lors, bien qu'elles demeurent en retrait pour la résolution de problèmes de taille importante, elles offrent une perspective des plus intéressante. Il est ainsi possible d'envisager une méthodologie basée sur la résolution exacte de sous-problèmes successifs du problème initial : c'est la recherche de grands voisinages, issue de paradigmes de la recherche locale.

Les approches de recherche locale sont en effet basées sur le principe fondateur suivant : à chaque itération, le voisinage d'une solution courante, contenant toutes les solutions atteignables à partir de celle-ci en réalisant un certain type de mouvement, est défini puis exploré à l'aide d'une méthode adaptée, dans l'optique de trouver une solution améliorante. Si une telle solution est découverte, alors le processus est itéré à partir de cette dernière. Au contraire, si l'examen du voisinage ne révèle aucune solution améliorante, alors la solution courante est un optimum local et plusieurs techniques peuvent alors être invoquées pour s'en extraire. Parmi les techniques les plus répandues figurent l'acceptation d'un voisin de moindre qualité, la modification du type de mouvement, la considération d'un voisinage plus étendu, etc.... Les méthodes de recherche locale alternent des phases pendant lesquelles une région donnée de l'espace de recherche est explorée minutieusement (intensification de la recherche) et des phases permettant de se diriger vers des régions inédites, délaissées auparavant par l'exploration (diversification de la recherche) [Glover 1997].

L'efficacité d'une méthode de recherche locale est fortement dépendante du voisinage qu'elle fait intervenir mais aussi de la façon qu'elle a de le parcourir, les deux aspects étant étroitement liés. Par exemple, un opérateur simple (1-opt, 2-opt) définira un voisinage de complexité polynomiale, aisément exploitable par une méthode énumérative mais cependant susceptible de conduire la recherche vers un optimum local dont il sera difficile de s'extraire, faute de mouvements suffisamment amples.

C'est pourquoi nous prenons le parti d'opter pour la considération d'un voisinage plus étendu. Deux types d'approches distinctes se placent dans ce contexte. Le premier vise à concevoir l'opérateur de voisinage de telle sorte que le problème de recherche de la meilleure solution voisine soit (pseudo-)polynomial, comme c'est par exemple le cas dans [Rios Solis 2005]. Une étude sur le sujet peut être trouvée dans [Ahuja 2002]. Nos travaux se situent dans le cadre d'une deuxième approche qui traite le cas où le problème de recherche du meilleur voisin est NP-difficile. Le voisinage est ici défini par la génération d'un sous-problème, que nous choisissons d'explorer à l'aide d'une procédure exacte, espérant de fait bénéficier des bonnes prestations de ce type de méthodes sur les problèmes de taille restreinte. En ce sens, la procédure de grand voisinage que nous allons maintenant présenter consiste en une coopération entre des paradigmes de recherche locale et exacte.

Cette forme de coopération, bien que moins répandue que les hybridations d'heuristiques désormais classiques, comme Genetic Local Search[2], tend tout de même à se démocratiser du fait des résultats compétitifs obtenus par ce genre d'approches sur des problèmes variés. Dans le cadre de problèmes d'ordonnancement, l'idée est même relativement ancienne. Ainsi, dès 1988, Adams

[2]qui consiste à appliquer une phase de recherche locale à certains des individus générés par un algorithme génétique

*et al.* [Adams 1988] proposaient leur procédure de *shifting bottleneck* pour la résolution de problèmes de job-shop. D'autres chercheurs leur ont alors emboîté le pas, au nombre desquels on peut compter Applegate et Cook [Applegate 1991] ou, plus récemment, Baptiste et al. [Baptiste 1995], ou encore Caseau et Laburthe [Caseau 1999a]. Aujourd'hui, le champ d'application ne se cantonne plus seulement aux problèmes d'ordonnancement et de nombreuses références peuvent être trouvées dans la littérature ([Shaw 1998], [Bent 2001], [Gendreau 2002], [Taillard 2002], ...). En outre, la recherche de grands voisinages n'est plus restreinte aux seules méthodes de recherche locale et s'intègre à présent au sein de processus arborescents exhaustifs ou non [Danna 2003a], [Beck 2003] ou de méthodes complètes [Verfaillie 1996].

De ces approches ressort néanmoins un dénominateur commun. C'est ce dernier que nous allons tenter d'extraire afin de présenter un schéma général d'unification, que nous appelons LSSPER pour Local Search (with) Sub-Problem Exact Resolution. Nous débutons ainsi cet exposé en section 1.4.2 par la présentation de ce schéma général d'exécution. Nous poursuivons par la spécification des différents points-clé de la méthode en sections 1.4.3, 1.4.4 et 1.4.5 et en émaillant notre propos de diverses références. En guise d'illustration d'un processus complet, nous terminons finalement par la présentation du schéma de branchement local (Local Branching [Fischetti 2002]) en section 1.4.6, et montrons la façon dont le formalisme présenté par LSSPER est adapté à ce cas particulier.

### 1.4.2 Description générale de la méthode

Étant donnée la résolution d'un problème général d'optimisation

$$P \ : \ \min_{X \in \mathcal{X}} f(X), X = (x_1, \ldots, x_n) \tag{1.1}$$

Durant tout ce mémoire, une lettre capitale (e. g. $X$) dénote un vecteur de variables de décision tandis qu'une lettre capitale surlignée (e. g. $\overline{X}$) représente un vecteur de valeurs des variables correspondantes.

L'approche de recherche de grands voisinages proposée ici s'inspire largement de la méthode MIMAUSA de Mautor et Michelon [Mautor 1997], [Mautor 2001]. Elle se propose d'alterner des phases d'intensification et de diversification [Glover 1997] dans la recherche des solutions.

Une phase d'intensification consiste à explorer profondément un sous-ensemble donné d'une région de l'espace de recherche $\mathcal{X}$ en résolvant des sous-problèmes successifs. Ainsi, à chaque itération $s$, un sous-problème $\Pi^s$ de taille $p$ est défini à partir de la solution courante $\overline{X}^{s-1}$ et des caractéristiques de $P$. $\Pi^s$ est ensuite résolu à l'aide d'une méthode appropriée et le résultat de l'optimisation est utilisé pour définir des stratégies de poursuite de la recherche, en particulier pour générer la solution $\overline{X}^s$ à partir de laquelle le processus est itéré.

Au contraire, le processus de diversification tend à visiter un sous-ensemble de la région $\mathcal{X}$ qui n'a pas encore été exploré jusqu'à présent. Il consiste à mettre en place des stratégies permettant d'orienter la recherche vers ces régions inédites.

Le schéma général d'exécution de LSSPER, qui retourne la solution finale $X^*$, est détaillé en Figure 1.6.

**Début**
1. Construire une solution initiale $\overline{X}^0$ au problème $P$
2. Initialiser $\overline{X}^*$, $s := 1$
3. Initialiser $p$
4. **Répéter**
5. Générer un sous-problème $\Pi^s$ de taille $p$ à partir de $P$ et de la solution courante $\overline{X}^{s-1}$
6. Résoudre $\Pi^s$ à l'aide d'une méthode appropriée
7. Adapter les stratégies de poursuite de la recherche au résultat de l'optimisation
8. Construire la solution $\overline{X}^s$
9. Mettre éventuellement à jour $\overline{X}^*$
10. **Si** la condition de diversification est vérifiée **Alors**
11. Mettre en place les stratégies de diversification
12. **Fin Si**
13. Ajuster éventuellement la valeur de $p$
14. $s := s + 1$
15. **Jusqu'à** l'atteinte d'un critère d'arrêt

**Fin**

FIG. 1.6 – *Algorithme général de la méthode proposée*

Le Pas 1 consiste à générer une solution initiale $\overline{X}^0$ pour $P$. Dans la majorité des cas, il s'agit de construire une solution complète à l'aide d'une heuristique adaptée au problème mais $\overline{X}^0$ peut également être une solution partielle voire la solution vide (racine de l'arbre de recherche par exemple). Les autres points majeurs de l'algorithme sont développés dans les sections suivantes :

- la génération d'un sous-problème et l'auto-ajustement de sa taille sont décrits en section 1.4.3,
- les stratégies de résolution d'un sous-problème et de poursuite de la recherche sont discutées en section 1.4.4,
- des procédures additionnelles d'intensification et de diversification sont présentées en section 1.4.5.

### 1.4.3 Une construction auto-adaptative du sous-problème

A chaque itération $s$, un sous-problème $\Pi^s$ de taille variable $p$ est construit en utilisant les caractéristiques du problème $P$ ainsi que sa solution courante $\overline{X}^{s-1}$ :

$$\Pi^s \ : \ \min_{Y \in \mathcal{Y}} g(Y), Y = (y_1, \ldots, y_p) \tag{1.2}$$

De façon générale, la technique employée revient à "geler" une partie de la solution courante $\overline{X}^{s-1}$. Globalement, la méthode de construction du sous-problème consiste à sélectionner $p \leq n$ variables $x_{i_1}, \ldots, x_{i_p}$ (renommées $y_1, \ldots, y_p$) de $P$, qui seront libres d'être réaffectées, et à fixer les $n - p$ variables restantes $x_{i_{p+1}}, \ldots, x_{i_n}$. Ces dernières peuvent être fixées de façon explicite aux valeurs $\overline{x}_{i_{p+1}}, \ldots, \overline{x}_{i_n}$ qu'elles occupent dans la solution courante, ou de façon implicite, par ajout

d'un ensemble de contraintes.

Adams et al. [Adams 1988] furent les premiers à incorporer cette technique à leur méthode de résolution de problèmes de job-shop : à chaque itération du processus, la machine constituant le goulet d'étranglement du processus est ordonnancée de façon optimale, avant d'être intégrée dans la solution courante. Le sous-problème correspond ici à un problème d'ordonnancement à une machine. D'autres travaux, également pour le problème de job-shop, parmi lesquels [Applegate 1991], ou encore [Caseau 1999a], s'inspirent de cette approche pour définir le sous-problème par la conservation de l'ordre de séquencement des opérations sur un sous-ensemble de $k$ (parmi $m$) machines. Toujours dans le domaine des problèmes d'ordonnancement, l'approche s'est vue généralisée à la conservation d'autres fragments de la solution courante : ensemble d'opérations, ou d'activités, s'exécutant dans une même fenêtre de temps [Caseau 1999a], [Palpant 2004], rangs d'exécution des opérations [Caseau 1999a], sous-ensemble aléatoire des ordres d'exécution relatifs des opérations nécessitant la même machine [Baptiste 1995], ... Des approches similaires ont également été appliquées à des problèmes d'affectation quadratique [Mautor 2001], de tournées de véhicules [Shaw 1998], [Gendreau 2002], [Bent 2001], ou encore d'affectation de fréquences [Palpant 2002]. Récemment, des applications à des problèmes linéaires en variables mixtes ont également été proposées. L'algorithme de branchement local [Fischetti 2002], par ajout d'une coupe spécifique au modèle originel, recherche des solutions voisines en imposant une limitation sur le nombre de variables pouvant être modifiées mais sans désigner à priori lesquelles le seront effectivement. La méthode de recherche de voisinages induits par relaxation (Relaxation Induced Neighbourhood Search) [Danna 2003a] consiste à fixer les variables qui possèdent la même valeur dans la relaxation continue courante et dans la meilleure solution. La recherche de grand voisinage est ici intégrée au sein d'un arbre de recherche, tout comme l'approche proposée par Beck et Refalo [Beck 2003].

Dans la plupart des approches énoncées ci-dessus, toute solution réalisable $\overline{Y}$ du sous-problème $\Pi^s$ est telle que $\overline{Z}$, vérifiant $\overline{z}_{i_1} = \overline{y}_1, \ldots, \overline{z}_{i_p} = \overline{y}_p$ et $\overline{z}_{i_{p+1}} = \overline{x}_{i_{p+1}}, \ldots, \overline{z}_{i_n} = \overline{x}_{i_n}$, est une solution réalisable pour $P$. Soient $\overline{Y}^*$, la solution optimale de $\Pi^s$ et $\overline{Z}^*$, son extension à $P$, la fonction objectif $g$ est alors telle que $f(\overline{Z}^*) \leq f(\overline{X})$.

Une alternative plus forte consiste à considérer des contraintes additionnelles telles que pour toute solution réalisable $\overline{Y}$ du sous-problème $\Pi^s$, $f(\overline{Z}) \leq f(\overline{X})$ [Palpant 2004]. Dans un tel cas, toute solution réalisable pour $\Pi^s$ peut trivialement être étendue afin d'obtenir un voisin $\overline{X}^s$ de $\overline{X}^{s-1}$ pour le problème $P$ vérifiant $f(\overline{X}^s) \leq f(\overline{X}^{s-1})$.

Cependant, certaines approches ne requièrent pas de telles assertions car elles utilisent la résolution de sous-problèmes à d'autres fins. C'est notamment le cas des méthodes qui intègrent une recherche de grands voisinages au sein d'un processus arborescent. Beck et Refalo [Beck 2003] proposent ainsi, pour un problème d'ordonnancement particulier, de résoudre à chaque nœud de l'arbre de recherche un sous-problème avec fenêtres de temps dont l'extension à une solution complète peut cependant se révéler impossible du fait des affectations de variables correspondant au nœud courant de l'arbre de recherche. Le but principal n'est pas ici de générer directement une solution complète à partir du résultat de l'optimisation d'un sous-problème mais d'en déduire des informations qui vont permettre d'accélérer la recherche ultérieure. L'algorithme des poupées russes [Verfaillie 1996], qui constitue également une approche complète, s'inscrit dans la même optique. Cette méthode consiste à résoudre des sous-problèmes de taille croissante à l'aide d'une procédure

arborescente classique.

Une autre approche, introduite par Pesant et Gendreau [Pesant 1999], consiste à résoudre un sous-problème en modélisant l'exploration du voisinage par des variables et contraintes. De façon générale, un modèle de voisinage est créé de telle façon que chaque solution réalisable du modèle représente un mouvement qui transforme la solution courante en une solution voisine. Un exemple simple revient à considérer le voisinage défini par l'échange des valeurs de deux variables $x_i$ et $x_j$. Ce dernier peut être parcouru par une double boucle sur les indices $i$ et $j$ mais également être défini par un modèle comportant deux variables $I$ et $J$ et une contrainte $I < J$, chaque solution réalisable $(i, j)$ correspondant à un échange unique. Le voisinage peut ainsi être parcouru à l'aide d'une procédure arborescente à laquelle il est possible d'appliquer les techniques habituelles de réduction. Lors de ce parcours, pour la recherche de la solution voisine, le modèle de voisinage ainsi que le modèle originel du problème sont tous deux actifs et communiquent par le biais de contraintes d'interface reliant les variables des deux modèles. De plus, une fonction objectif peut être définie, afin de rechercher le meilleur voisin. Cependant, si cette approche présente des avantages certains, elle n'est cependant réellement intéressante que lorsque les techniques spécifiques de recherche arborescente permettent une réduction significative de la taille du voisinage.

### Discussion

La question essentielle des méthodes de recherche de grands voisinages réside dans la définition du sous-problème, c.-à-d. les variables qu'il convient de sélectionner simultanément. Celle-ci résulte généralement d'une analyse dépendante du problème des contraintes de $P$ tout en conservant, idéalement, les objectifs suivants à atteindre :

- le voisinage défini par $\Pi^s$ doit être de taille raisonnable, c.-à-d. qu'il doit contenir un grand nombre de solutions tout en demeurant exploitable par une méthode d'énumération implicite ou une heuristique ;
- le voisinage défini par $\Pi^s$ contient des solutions prometteuses ;
- la région réalisable $\mathcal{X}$ de $P$ est suffisamment explorée en résolvant des sous-problèmes consécutifs $\Pi^q$, $q \geq 1$ (diversification de la recherche) : les $p$ variables doivent être sélectionnées de façon non-uniforme.

Une première règle qui semble émerger quant à la définition d'un voisinage "prometteur" repose sur la libération simultanée de variables reliées les unes aux autres. Deux raisons à cela : premièrement, si l'on se contente de libérer des variables de façon aléatoire, le problème résiduel peut demeurer suffisamment contraint pour que l'unique extension possible de la solution parcellaire gelée soit la solution actuelle. Il est alors essentiel de libérer des variables possédant des liens entre elles afin qu'elles s'octroient les unes les autres des possibilités de réaffectation. Deuxièmement, l'esprit de la recherche de grands voisinages est de considérer des sous-problèmes qui ne sont pas la concaténation de sous-problèmes indépendants plus petits mais des problèmes à part entière, dont la résolution conduit vers un gain plus important que des réaffectations de variables réalisées de façon isolée. La libération simultanée de variables liées permet ainsi à chacune d'elle de prendre des valeurs plus pertinentes que si elles étaient modifiées de façon indépendante. Dans la littérature, le choix de variables dépendantes est souvent réalisé grâce à la connaissance de la structure de haut

niveau du problème considéré.

Une deuxième piste concerne l'auto-ajustement de la taille du voisinage. Celui-ci dépend bien évidemment de la quantité d'information conservée : un sous-problème de taille modeste (une faible valeur de $p$) induira un voisinage de taille restreinte qui sera parcouru aisément par une méthode complète mais susceptible de faire converger la recherche vers un optimum local, possiblement très éloigné de l'optimum global, dont il sera très difficile de s'extraire ; une valeur plus élevée de $p$ permettra quant à elle de considérer un voisinage plus étendu mais dont l'exploration peut dans certains cas s'avérer extrêmement coûteuse. Il s'agit alors de trouver un compromis entre la taille du voisinage et le temps dispensé pour son parcours, ce qui est réalisé dans nombre d'approches par un ajustement automatique de la valeur de $p$. Dans [Danna 2003a], la taille du sous-problème est incrémentée dès qu'un nombre donné d'échecs d'optimisation consécutifs est rencontré jusqu'à atteindre une limite supérieure $\overline{p}$. Dans [Mautor 2001], [Palpant 2003a] et [Palpant 2004], l'auto-ajustement de la taille $p$ est lié à des considérations sur les temps dispensés par la méthode de résolution des sous-problèmes : si ce temps tend à s'accroître, alors $p$ est décrémenté, jusqu'à ce qu'une borne inférieure $\underline{p}$ soit atteinte ; au contraire, si le temps de résolution tend à diminuer, la taille du sous-problème est augmentée, jusqu'à ce que $p = n$. Dans [Fischetti 2002], l'incapacité de trouver une solution améliorante induit des modification de la taille de $p$, ainsi que nous le verrons de façon précise en section 1.4.6.

Finalement, un dernier point concerne la non-uniformité du schéma de sélection du sous-problème. Ce dernier point, s'il peut s'avérer superflu lorsque l'exploration de l'espace de recherche est réalisée de façon systématique ([Beck 2003], [Verfaillie 1996], [Fischetti 2002], ...), est crucial dans tous les autres cas. Il permet en effet de visiter potentiellement des régions de l'espace de recherche inédites. En effet, lorsque le processus de sélection est déterministe, il n'offre la possibilité de générer à partir d'une solution courante donnée qu'un unique sous-problème qui, s'il échoue à être optimisé, conduit à une impasse dont il est alors nécessaire de s'extraire par d'autres moyens [Mautor 2001]. Il peut alors s'avérer intéressant de biaiser ce processus de sélection à l'aide d'une composante aléatoire [Palpant 2003a], [Palpant 2004].

### 1.4.4 Résolution du sous-problème et stratégies de poursuite de la recherche

Une fois défini, un essai de résoudre le sous-problème $\Pi^s$ est réalisé. Le plus souvent, la procédure consiste en une recherche arborescente tronquée. Le paramètre de troncature peut être une limite sur le nombre de nœuds [Danna 2003a], un temps d'exécution maximal [Fischetti 2002], [Palpant 2003a], [Palpant 2004], ou bien encore une limite de divergences [Caseau 1999a]. Une valeur élevée du paramètre de limitation correspond à une intensification de la recherche car plus de moyens sont alors alloués pour la recherche de la meilleure solution de $\Pi^s$, c.-à-d. du meilleur voisin. Inversement, instancier le paramètre de limitation à une valeur faible revient à espérer trouver rapidement une solution réalisable pour $\Pi^s$.

Des méthodes de résolution appropriées, liées aux caractéristiques du sous-problème, peuvent également être employées. Ainsi, la procédure de shifting bottleneck [Adams 1988] résout successivement des problèmes d'ordonnancement à une machine pour lesquels un algorithme très per-

formant [Carlier 1982] est disponible. De même, la procédure shuffle [Applegate 1991] utilise un algorithme approprié utilisant un procédé de propagation de contraintes (edge-finding) pour la résolution des sous-problèmes.

Dès que le sous-problème $\Pi^s$ a été optimisé à l'aide de la procédure choisie, des stratégies de poursuite de la recherche sont mises en place en concordance avec le résultat de l'optimisation. Il s'agit tout d'abord de générer la nouvelle solution $\overline{X}^s$ à partir de laquelle le processus sera poursuivi.

Si une solution $\overline{Y}^s$ au sous-problème $\Pi^s$ a été obtenue, $\overline{X}^s$ peut être obtenue en étendant $\overline{Y}^s$ au problème global $P$. La forme d'extension la plus triviale, lorsqu'elle est possible, consiste à remplacer dans $\overline{X}^{s-1}$ les valeurs $\overline{x}_{i_1}, \ldots, \overline{x}_{i_p}$ par celles trouvées lors de la résolution de $\Pi^s$, à savoir $\overline{y}_1, \ldots, \overline{y}_p$. La solution $\overline{X}^s$ ainsi obtenue constitue alors un voisin, dans le sens recherche locale du terme, de la solution $\overline{X}^{s-1}$. Cependant, une telle extension n'est pas toujours possible et d'autres moyens peuvent être mis en oeuvre pour générer $\overline{X}^s$. Dans [Adams 1988], la solution courante (partielle) est réajustée en résolvant une série de problèmes à une machine sur les machines critiques. Dans [Palpant 2004], une procédure de post-optimisation est appliquée à l'extension triviale de $\overline{Y}^s$. Dans [Beck 2003], une procédure de PPC simple est utilisée pour étendre la solution $\overline{Y}^s$, obtenue à un nœud donné, au problème global. En cas d'échec, le processus arborescent continue normalement, le résultat de l'optimisation du sous-problème étant utilisé pour mettre à jour des informations sur les bornes. Si la procédure d'extension se révèle fructueuse cependant, ou si aucune solution n'est trouvée au sous-problème, le sous-arbre situé sous le nœud courant est écarté et l'exploration est poursuivie. La recherche de grand voisinage active ainsi la fermeture additionnelle de certains nœuds, directement ou par l'intermédiaire de la borne qu'elle contribue à mettre à jour, et permet de fait une accélération du processus global, mais n'influe que de façon très indirecte sur la détermination de la nouvelle solution (partielle) $\overline{X}^s$. Dans les méthodes de recherche arborescente, cette dernière est en effet obtenue par un déplacement vers un nouveau nœud de l'arbre de recherche dicté par les stratégies de parcours mises en oeuvre (Probe [Beck 2003], guided dives [Danna 2003a]). Dans [Verfaillie 1996], la recherche est redémarrée à partir d'une solution vide (associée à la racine de l'arbre servant à la recherche de la solution du sous-problème suivant). Il s'agit, à la manière des poupées russes, d'emboîter les sous-problèmes les uns dans les autres, en résolvant à chaque itération un sous-problème comportant une variable supplémentaire par rapport au sous-problème précédent. L'idée est ici que la résolution d'un sous-problème ne guide la suite du processus que par l'intermédiaire de la mise à jour d'informations sur les bornes, en espérant que celles-ci conduisent à l'élimination d'un grand nombre de nœuds pour les résolutions ultérieures. Cette méthodologie n'est applicable que par un choix et un enchaînement dans la résolution des sous-problèmes bien déterminés. D'autres types d'informations complémentaires peuvent également être tirées du résultat de l'optimisation. Par exemple, dans [Fischetti 2002], la coupe de branchement est renversée et ajoutée au modèle originel dès lors que l'optimisation du sous-problème est complète (limite de temps non atteinte) afin d'interdire la re-visite de régions déjà complètement explorées.

### 1.4.5 Intensification et diversification additionnelles de la recherche

En dépit du fait que la taille importante des voisinages considérés ainsi que les méthodes puissantes mises en oeuvre pour la résolution des sous-problèmes procurent de façon implicite des propriétés de diversification et d'intensification de la recherche, respectivement, d'autres mécanismes peuvent être mis en place afin d'amplifier davantage l'un ou l'autre des aspects.

Des procédés d'intensification issus du domaine de la recherche locale classique peuvent ainsi être employés, mais de nouvelles stratégies, directement reliées aux spécificités de la méthode, peuvent également être mises en place : considération d'un voisinage plus étendu (par augmentation de la taille du sous-problème), allocation d'une limite de résolution du sous-problème plus importante, modification de la stratégie de sélection des variables, définition de stratégies systématiques d'orientation de la recherche vers les voisinages de solutions de qualité (guided dives [Danna 2003b]), ...

De la même façon, des stratégies de diversification additionnelle peuvent être instaurées. Il peut ainsi s'agir d'accepter une solution voisine non améliorante [Fischetti 2002], [Mautor 2001], redémarrer la recherche à partir d'une nouvelle solution initiale [Palpant 2004], considérer des voisinages différents [Caseau 1999a], dans l'esprit de la recherche de voisinage variable (Variable Neighbourhood Search [Mladenović 1997]), ...

### 1.4.6 Exemple d'application du schéma général proposé par LSSPER : Local branching

Fischetti et Lodi [Fischetti 2002] présentent une méthode de résolution de programmes linéaires en variables mixtes (MIP) qu'ils dénomment branchement local (Local Branching). Sa version heuristique, à laquelle nous nous intéressons, s'inscrit parfaitement dans le cadre général proposé par LSSPER.

A chaque itération $s$ un sous-problème $\Pi^s$ est généré à partir de la solution courante $\overline{X}^{s-1}$ en ajoutant au système de contraintes originel une coupe dite de branchement local : $\Delta(X, \overline{X}^{s-1}) \leq p$. Cette dernière impose la recherche de solutions situées au maximum à une distance $p$ donnée de la solution courante. Dans le cas de MIP, cette distance correspond au nombre de variables binaires dont la valeur est complémentée mais on peut fort bien imaginer d'autre mesures de distance pour des problèmes non mixtes.

Le sous-problème est alors résolu à l'aide d'une procédure arborescente tronquée au bout d'un certain temps d'exécution $H$. Les différents résultats possibles sont reportés en figure 1.7 : en partie gauche, la méthode exacte ne dépasse pas la limite de temps qui lui a été allouée, en partie droite oui ; en partie supérieure, une solution améliorante est retournée au terme de l'exécution, en partie inférieure non.

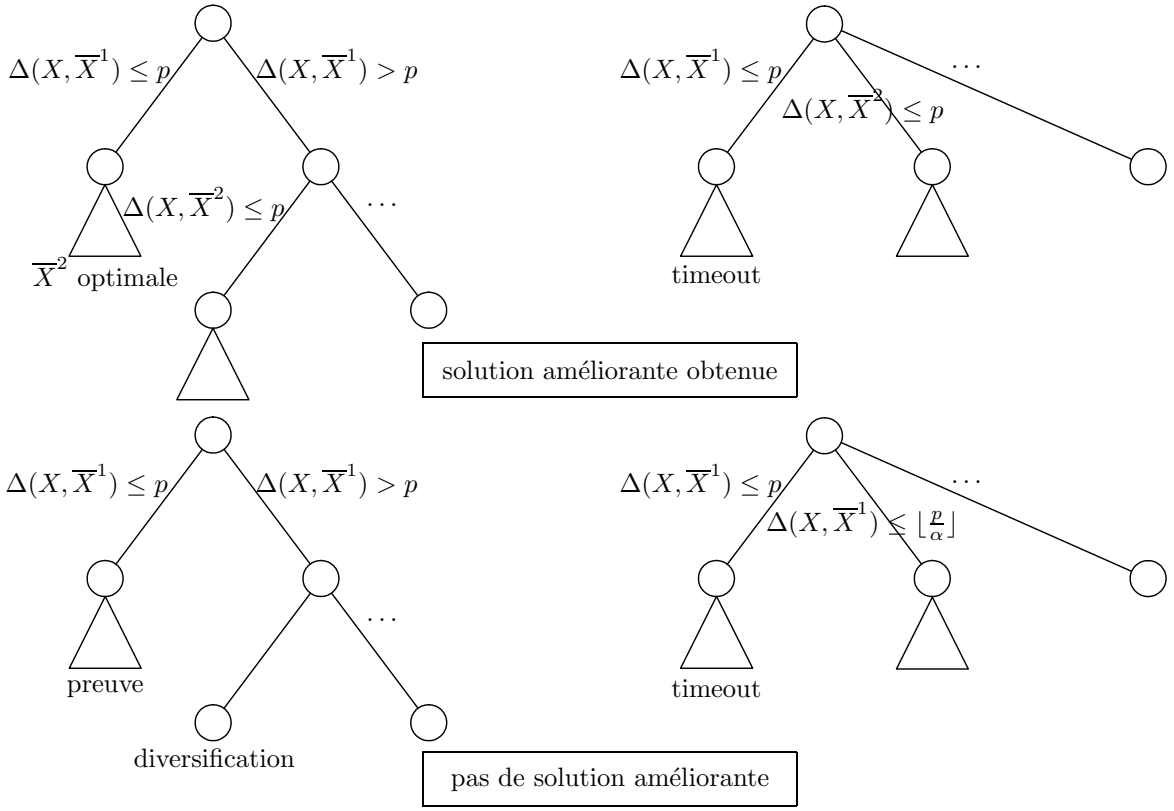


FIG. 1.7 – *Principes de fonctionnement de Local Branching*

Le résultat de l'optimisation de $\Pi^s$ est alors utilisé pour guider la suite de la recherche. Dans le cas où une solution améliorante a été obtenue, celle-ci constitue la nouvelle solution $\overline{X}^s$ à partir de laquelle le processus est itéré. En cas d'échec, la nouvelle solution est obtenue en posant $\overline{X}^s = \overline{X}^{s-1}$. Lorsque la limite de temps n'a pas été dépassée, la coupe de branchement local est renversée : $\Delta(X, \overline{X}^{s-1}) > p$ ; et vient enrichir le système de contraintes originel du problème afin d'empêcher la recherche de se diriger à nouveau vers une région déjà complètement explorée et donc exploitée de façon optimale. Dans cette condition, et lorsqu'aucune solution n'a été en mesure d'améliorer la solution courante, une phase de diversification est mise en place. Celle-ci consiste alors à rechercher une solution améliorante dans un voisinage plus grand (diversification douce), ou à rechercher la première solution non améliorante dans un voisinage plus grand encore (diversification forte). Les tailles de voisinages sont ici reliées à la valeur de $p$ qui, comme dans le cadre général proposé par LSSPER, s'ajuste aux besoins :

- diversification douce : $p = \lfloor \beta p \rfloor, \beta > 0$
- diversification forte : $p = \lfloor \beta^2 p \rfloor, \beta > 0$
- temps limite atteint et aucune solution retournée : $p = \lfloor \frac{p}{\alpha} \rfloor, \alpha > 0$

## 1.5 Conclusion

Nous avons présenté au cours de ce chapitre une certaine forme de coopération entre paradigmes issus de recherche exacte et heuristique. Nous nous sommes ainsi intéressés aux méthodes intégrant les concepts liés à ces deux approches au sein d'un processus global. Le but de la démarche consiste à tirer parti des caractéristiques complémentaires de chacune des approches : systématicité et optimalité de la composante exacte, caractère moins déterministe et rapidité de la composante heuristique. Nous avons constaté à travers notre étude que ces méthodes hybrides couvrent un large éventail d'applications et leurs performances de plus en plus reconnues font d'elles un sujet d'études très prisé.

Nous allons présenter au cours des deux chapitres suivants des adaptations de LSSPER pour la résolution de deux problèmes bien distincts : d'une part, le problème d'ordonnancement de projet sous contraintes de ressources, d'autre part, le problème d'affectation de fréquences avec sommation des perturbateurs. Les résultats obtenus sur ces deux types de problèmes semblent exhiber l'efficacité de notre approche de recherche de grand voisinage. Nous développerons ensuite au cours du dernier chapitre une approche de type recherche exacte incomplète. Cette dernière consiste en l'hybridation de la méthode de Resolution Search avec des paradigmes heuristiques.

# Chapitre 2

# Application de LSSPER au Problème d'Ordonnancement de Projet sous Contraintes de Ressources

*Ce chapitre présente en détail l'application de LSSPER au problème d'ordonnancement de projet sous contraintes de ressources, auquel nous ferons référence dans la suite de notre propos par son acronyme anglais RCPSP, pour Resource-Constrained Project Scheduling Problem. Le choix de ce problème réside dans le fait qu'il constitue un problème académique étudié par une pléiade d'auteurs au cours des dernières années. Ainsi, la profusion de méthodes ainsi que la disponibilité de nombreux jeux de données nous ont permis d'éprouver de façon sérieuse l'efficacité de la méthode que nous proposons. Dans une première partie (2.1), nous présentons une description du problème. Puis nous nous attardons sur les différentes recherches qui ont été menées sur le sujet en nous focalisant sur les méthodes heuristiques et metaheuristiques de résolution de ce problème (2.2). Nous poursuivons par la spécification de notre méthode de grand voisinage pour la résolution de ce problème particulier (2.3) et exhibons les résultats expérimentaux obtenus sur divers jeux d'instances classiques. Nous clôturons finalement ce chapitre par la présentation d'un schéma de résolution en cascade permettant une amélioration significative de l'efficacité de la méthode (2.4). Ces travaux ont fait l'objet de publications dans [Palpant 2003a] et [Palpant 2004].*

## 2.1 Description du RCPSP

### 2.1.1 Définition

Le RCPSP s'intéresse à ordonnancer l'exécution d'un ensemble $A$ de $n$ *activités* (ou *tâches*), liées les unes aux autres par des *contraintes de précédence*, sur un ensemble $R$ de $m$ *ressources*, renouvelables et cumulatives. Plus formellement, le RCPSP consiste à caractériser le n-uplet $S = \{S_1, \ldots, S_n\}$ des *dates de début* d'exécution des activités du projet tout en respectant un certain nombre de contraintes. Chaque activité du projet est *non-interruptible* (ou *non-préemptive*) durant toute la durée de son exécution $p_i$. Ainsi, la date de fin d'exécution d'une activité $i$ est donnée par $C_i = S_i + p_i$. Les contraintes de précédence simples entre activités définissent une relation qui peut être représentée par un graphe orienté acyclique $(A, P)$ qui impose à toute activité $j \mid (i,j) \in P$, de ne pas débuter son exécution avant la complétion de l'activité $i$. On note par $\Gamma_i^{-1}$ (resp. $\Gamma_i$) l'ensemble des prédécesseurs (resp. successeurs) de l'activité $i$. Une instance du RCPSP est usuellement modélisée par un graphe $G = (V, E, p)$, valué, orienté et acyclique, le *graphe potentiel-tâches*, dont l'ensemble des sommets est l'ensemble $A$ augmenté de deux tâches fictives 0 et $n+1$ de durées nulles, représentant respectivement le début et la fin du projet, telles que $S_0 = 0$ et $S_{n+1} \geq C_i \ \forall i \in A$. L'ensemble des arcs de G est donné par $E = P \cup \{(0,i) \mid i \in A, \Gamma_i^{-1} = \emptyset\} \cup \{(i, n+1) \mid i \in A, \Gamma_i = \emptyset\}$.

Enfin, chaque activité $i$ monopolise au cours de son exécution une quantité positive ou nulle $r_{ik}$ (appelée *consommation*) d'unités de chaque ressource $k$. Cette quantité redevient disponible une fois l'activité terminée et l'on parle alors de ressources *renouvelables*. Les ressources sont également *cumulatives*, c'est-à-dire qu'elles permettent l'exécution simultanée de plusieurs activités, exécution qui est assujettie au respect des limitations de ressources. Ainsi, chaque ressource $k$ possédant une *capacité* constante $R_k$ tout au long du projet, la consommation totale de la ressource par les activités en cours d'exécution ne doit-elle pas excéder cette valeur quel que soit l'instant $t$ considéré.

L'objectif étudié dans le cadre de cette thèse est la minimisation de la durée totale du projet (ou *makespan*) $S_{n+1}$, dont une borne supérieure $UB$ est donnée. Le RCPSP peut alors être modélisé de la façon suivante :

$$(P) \qquad \min S_{n+1} \tag{2.1}$$

$$S_j \geq S_i + p_i \qquad \forall (i,j) \in E \tag{2.2}$$

$$\sum_{i \in \mathcal{A}(\tau, S)} r_{ik} \leq R_k \quad \forall k \in R, \forall \tau \in \{0, \ldots, UB\} \tag{2.3}$$

où $S_0 = 0$ et $\mathcal{A}(\tau, S)$ représente l'ensemble des activités en cours d'exécution à l'instant $\tau$, c.-à-d. vérifiant $S_i \leq \tau < S_i + p_i$.

A partir des données de ce problème, nous pouvons définir les notations suivantes qui seront utilisées dans la suite de ce chapitre :

- $ES_i$ (resp. $EF_i$) : "*Earliest Starting Time*" (resp. "*Earliest Finishing Time*") ou date de début (resp. de fin) *au plus tôt* de l'activité $i$. Elle représente la date avant laquelle $i$ ne peut commencer (resp. s'achever). En l'absence de contraintes de ressources, $ES_i$ est équivalente à la longueur du plus long chemin de 0 à $i$ dans $G$.

– $LS_i$ (resp. $LF_i$) : "*Latest Starting Time*" (resp. "*Latest Finishing Time*") ou date de début (resp. de fin) *au plus tard* de l'activité $i$. Elle représente la date après laquelle $i$ ne peut commencer (resp. s'achever). Si l'on supprime les contraintes de ressources, $LS_i$ est égale à la différence entre la borne supérieure $UB$ et la longueur du plus long chemin entre $i$ et $n+1$ dans G.

Par définition, $ES_{n+1}$ correspond à la durée minimale du problème lorsque l'on occulte les contraintes de ressources. En leur présence, ce problème est NP-difficile au sens fort (voir Blazewicz et al. [Blazewicz 1983]).

### 2.1.2 Exemple

Un exemple simple d'une instance à 10 tâches et une ressource unique de capacité 4 est donné dans [Klein 1999] et peut se représenter par le graphe potentiel-tâches correspondant (partie supérieure de la figure 2.1). En dessous de chaque nœud représentant une activité $i$, on indique sa durée $p_i$ et sa consommation $r_{i1}$. En partie inférieure de la figure, un ordonnancement réalisable de durée totale égale à 24 est modélisé par le diagramme de Gantt d'occupation de la ressource.

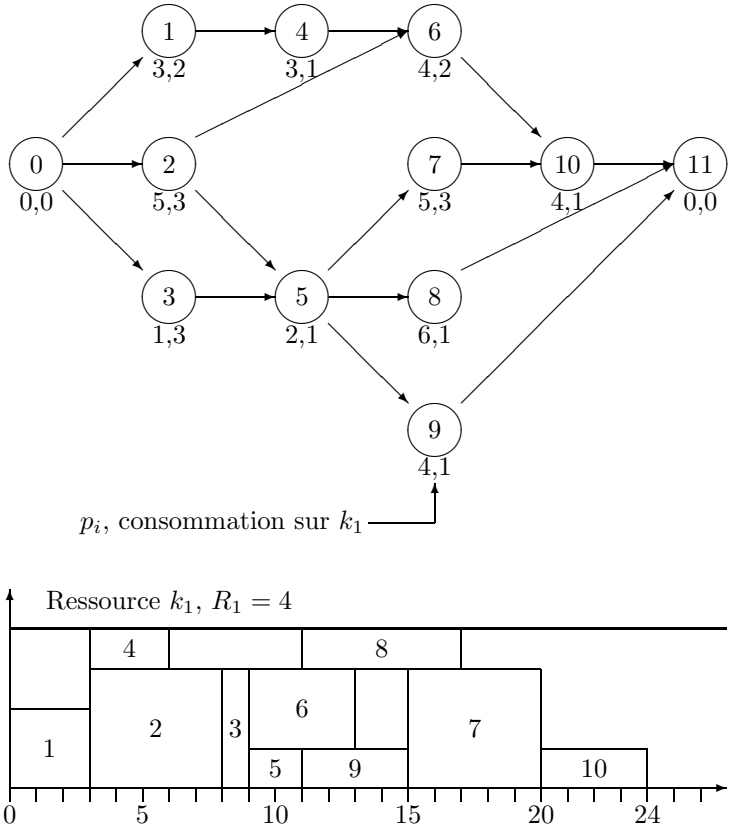


FIG. 2.1 – *Exemple d'instance de RCPSP*

### 2.1.3 Applications et problèmes connexes

Le RCPSP, de par la variété d'applications qu'il recouvre, a été l'objet de nombreuses recherches. Ainsi, de multiples problèmes industriels ou relevant du domaine organisationnel peuvent être modélisés comme un problème de RCPSP :

- les problèmes de découpes : des matériaux sont disponibles en rouleaux de largeur standard $b$, et doivent être découpés en $n$ rouleaux de largeurs respectives $a_1, \ldots, a_n$. Le problème consistant à déterminer le nombre de rouleaux standards nécessaires pour détailler les rouleaux $i = 1, \ldots, n$ ainsi les découpes à réaliser peut être vu comme un problème de RCPSP à une ressource renouvelable de capacité $R_1 = b$ et $n$ activités, chaque activité $i$ s'exécutant durant 1 unité de temps et requérant $a_i$ unités de la ressource. Le makespan correspond ici au nombre de rouleaux standards à découper.
- les problèmes d'ateliers classiques : *flow-shop*, *job-shop*, *open-shop*, qui consistent à ordonnancer un ensemble de travaux, décomposés en une succession d'opérations à réaliser sur des machines *disjonctives*, c.-à-d. de capacité unitaire (on parle de *gamme* opératoire). Les séquences d'opérations peuvent être prédéfinies, comme pour le flow-shop, où tous les travaux utilisent les machines dans le même ordre, ou le job-shop, qui associe à chaque travail une gamme de fabrication propre, ou non. C'est le cas de l'open-shop, pour lequel les travaux n'ont pas de gamme fixée et peuvent ainsi utiliser les machines dans un ordre quelconque. Dans le cas où la gamme est déterminée, celle-ci définit une relation de précédence entre les opérations constitutives d'un travail donné. Tous ces problèmes peuvent être considérés comme des cas particuliers de RCPSP, où les opérations correspondent aux activités tandis que les machines et travaux peuvent être entrevus comme des ressources. La disjonction au niveau de l'utilisation d'une ressource (machine ou travail) est réalisée en imposant sa consommation totale par les activités qui la requièrent au cours de leur exécution (consommations et disponibilités des ressources unitaires).
- les problèmes d'ateliers avec étages : *flow-shop hybride*, *job-shop et open-shop généralisés*, qui étendent les problèmes d'atelier classiques en mettant à disposition à chaque étage un ensemble de machines identiques pour la réalisation des opérations. Ainsi, il s'agit de déterminer l'affectation d'une machine à chaque opération en même temps que de séquencer les opérations sur les machines. Ces problèmes sont également des RCPSP qui peuvent être définis en étendant la formulation précédente à des ressources de capacité non unitaire. Ainsi, chaque étage est associé à une ressource dont la capacité est égale au nombre de machines qui y sont disponibles. Les consommations réalisées par les activités (opérations) sur les ressources (machines et travaux) qui leur sont associées demeurent unitaires.
- les problèmes d'emploi du temps qui consistent à planifier un nombre donné d'activités impliquant divers groupes de personnes et nécessitant des ressources matérielles et/ou spatiales. On peut en particulier se référer au problème simple d'emploi du temps à l'université, où il s'agit de répartir les cours par enseignants et par salles. Il s'agit alors de résoudre un problème de RCPSP au sein duquel les activités représentent les cours à planifier et les ressources unitaires les salles et enseignants disponibles.

Il apparaît cependant que la formulation classique du RCPSP n'est parfois pas suffisante pour décrire certains aspects de la vie réelle. C'est pourquoi de nombreuses variantes de ce problème ont vu le jour afin de permettre l'intégration de contraintes additionnelles. Il peut alors s'agir d'autoriser la préemption des activités ou encore de considérer le cas de ressources non-renouvelables. Il s'avère également parfois nécessaire de généraliser les contraintes ou encore définir de nouveaux critères d'optimisation.

Les données peuvent ainsi varier en fonction du mode ou du temps. C'est notamment le cas du RCPSP multi-mode, qui autorise des modes distincts d'exécution pour une même activité, correspondant chacun à des durées et consommations de ressources propres. Les alternatives peuvent porter sur un rapport temps/consommation (ex : 1 unité de ressource pendant 5 heures ou 5 unités pendant 1 heure) ou encore ressource/ressource (ex : à temps d'exécution égal, 2 unités sur une ressource ou 1 unité sur 2 ressources). Capacités et consommations peuvent également varier en fonction du temps. Le cas de capacités variables consiste en une généralisation des ressources renouvelables et/ou non-renouvelables : on considère alors les ressources comme partiellement renouvelables, c.-à-d. non renouvelables sur des intervalles de temps donnés. Dans le cadre de consommations variables, le problème s'apparente alors au LCSP (Labour-Constrained Scheduling Problem).

La généralisation des contraintes de précédence, $S_j - S_i \geq d_{ij}\ \forall (i,j) \in V \times V$, permet d'introduire à la fin de l'exécution d'une activité $i$ un délai avant la fin duquel l'activité $j$ ne pourra débuter. Ces contraintes sont requises notamment dans le cas de problèmes d'ordonnancement avec temps de préparation (dépendants ou non de la séquence).

Le travail présenté dans ce mémoire s'intéresse exclusivement à la résolution du problème classique de RCPSP. Cependant, dans le cadre de la méthode développée, nous serons également amenés à résoudre des variantes de RCPSP à ressources partiellement renouvelables et fenêtres de temps (cf. section 2.3.3).

## 2.2 Schémas d'ordonnancement, metaheuristiques et voisinages

Dans cette partie, nous allons passer en revue les principales heuristiques et metaheuristiques ayant été appliquées au RCPSP ou à des problèmes connexes (job-shop notamment). Cet état de l'art se veut non exhaustif et nous recommandons la lecture de [Kolisch 2004] et [Vaessens 1996] pour une vision plus complète. Nous nous focaliserons en particulier sur les méthodes récentes ayant obtenus des résultats très compétitifs sur divers jeux d'instances.

Ainsi, nous prendrons le parti de mettre en lumière pour chacune des approches classiques de résolution heuristique les méthodes nous apparaissant comme étant les plus représentatives de l'approche considérée.

En section 2.2.1, nous débutons notre revue par la présentation d'heuristiques constructives basées sur des schémas de génération d'ordonnancement, ou SGS pour Schedule Generation Scheme. Ces premières heuristiques sont souvent utilisées comme procédures élémentaires dans des méthodes de résolution plus complexes. La section 2.2.2 est consacrée aux algorithmes d'ordonnancement à

passes multiples. Nous détaillons en particulier le fonctionnement d'un algorithme de *forward-backward*. Les algorithmes évolutionnistes (algorithmes génétiques et colonies de fourmis) seront présentés en section 2.2.3. La section 2.2.4 traite quant à elle des algorithmes de recherche locale (recherche taboue, recuit simulé). Nous terminons finalement cette revue de la littérature en abordant le cas des algorithmes hybrides en section 2.2.5. Nous étudions en particulier les méthodes faisant cohabiter résolution exacte et processus heuristique.

### 2.2.1 Heuristiques constructives et SGS

Les heuristiques *constructives*, de type glouton, construisent une solution réalisable à partir d'un ordonnancement vide, en choisissant à chaque itération une activité parmi un ensemble d'*activités candidates* et en l'insérant dans l'ordonnancement courant. L'ensemble des activités candidates à une itération donnée ainsi que le processus d'ordonnancement d'une activité sont tous deux déterminés par un SGS, le choix de l'activité à ordonnancer résultant quant à lui de l'application d'une *règle de priorité*. L'algorithme s'achève lorsque toutes les activités ont été traitées. Ainsi, de telles méthodes, comportant un nombre d'itérations équivalent au nombre d'activités dans le projet, sont-elles très performantes en terme de temps d'exécution. Elles sont appelées heuristiques *à passe unique* car elles réalisent un seul appel à un SGS.

Deux schémas de génération sont couramment utilisés : le SGS *sériel* et le SGS *parallèle*. Le SGS sériel, utilisé dans la présente étude, détermine à chaque itération les activités candidates comme l'ensemble des activités non encore ordonnancées dont tous les prédécesseurs ont déjà été insérés dans l'ordonnancement. L'activité sélectionnée à l'aide de la règle de priorité choisie est ensuite positionnée à la date de début la plus petite permise par les contraintes de précédence et de ressources imposées par les activités préalablement ordonnancées : les dates de début d'exécution des activités sont calées au plus tôt. L'*ordonnancement au plus tôt* ainsi généré nécessite l'exécution d'une *passe avant* du SGS.

De façon symétrique, il est possible de bâtir un ordonnancement à partir d'un instant de référence correspondant à la fin du projet et en ordonnançant les activités antérieurement à cette date. A chaque itération, l'ensemble des activités candidates est ici constitué de toutes les activités non encore ordonnancées dont tous les successeurs ont préalablement été traités. De même, l'activité prioritaire par rapport à la règle utilisée, est-elle insérée dans l'ordonnancement à la date de début la plus grande permise par les contraintes de succession et de ressources découlant du positionnement des activités précédemment ordonnancées : les dates de début d'exécution des activités sont calées au plus tard. L'*ordonnancement au plus tard* ainsi déterminé nécessite l'exécution d'une *passe arrière* du SGS.

Les algorithmes basés sur l'utilisation d'un SGS au cours d'une passe avant ou arrière ont une complexité de $\theta(n^2m)$.

L'ensemble des ordonnancements pouvant être générés à l'aide du SGS sériel est appelé l'ensemble des ordonnancements *actifs*. Cet ensemble possède des propriétés de *dominance* en regard du critère d'optimisation (minimisation du makespan). Nous référons à [Kolisch 1999] pour une description plus précise des SGS sériel et parallèle.

### 2.2.2 Les algorithmes à passes multiples

Les avantages liés à la rapidité d'exécution des heuristiques constructives sont souvent malheureusement annihilés par leurs performances médiocres en regard de la qualité des solutions obtenues. De nouvelles heuristiques ont ainsi vu le jour, dont le principe est de pallier la déficience qualitative des heuristiques constructives en relançant de multiples fois l'exécution de ces dernières. Ce sont les heuristiques *à passes multiples*.

Une première approche consiste à itérer le processus constructif en utilisant lors de chaque exécution une règle de priorité différente afin de générer des ordonnancements distincts. Dans le même esprit, le caractère systématique de l'utilisation d'une même règle de priorité au cours des exécutions successives, conduisant inéluctablement à la production d'un unique ordonnancement, peut être éliminé par l'insertion d'une composante aléatoire permettant de biaiser la sélection (random biased sampling). Nous référons à [Hartmann 2000] et [Schirmer 2000] pour plus de détails concernant ce type de méthodes.

Une deuxième approche, initiée par Li et Willis [Li 1992], consiste à bâtir des ordonnancements en faisant appel à un SGS au cours de passes avant et arrière alternées : ce sont les heuristiques de *forward-backward*. Au cours chaque passe, un SGS est utilisé conjointement à une règle de priorité adaptée.

L'approche envisagée par Li et Willis préconise l'utilisation du schéma sériel. Les ordonnancements successifs sont obtenus de la façon suivante : au cours d'une passe avant, les activités sont ordonnancées à partir de l'instant de référence $t = 0$ selon la procédure habituelle ; au cours d'une passe arrière, les activités sont ordonnancées antérieurement à la date de référence constituée par le makespan obtenu lors de la passe avant précédente. Cependant, les règles de priorité utilisées ne permettent pas d'assurer la construction d'un ordonnancement réalisable. Il arrive en effet fréquemment que l'exécution de certaines activités doive être, au moins partiellement, réalisée dans le passé. Afin de contrer cet effet néfaste, d'autres heuristiques ont pris le parti d'utiliser des règles de priorité ou des mécanismes de construction d'ordonnancements adaptés. C'est ainsi le cas de la méthode d'Özdamar et Ulusoy [Özdamar 1996], qui génère tous les ordonnancements à partir de l'instant $t = 0$ en employant le schéma parallèle. L'utilisation conjointe d'une procédure d'évaluation et de résolution de conflits de ressources entre activités (LCBA, pour Local Constraint Based Analysis), qui réalise la mise à jour des fenêtres de temps d'exécution des activités et de la durée du projet, assure la production d'ordonnancements réalisables. C'est également le cas de la procédure mise au point par Tormos et Lova [Tormos 2001] qui sera reprise sous le terme de "double justification" par Valls *et al.* [Valls 2004a]. Cette technique d'amélioration applique deux passes successives à un ordonnancement existant : les activités sont d'abord décalées vers la droite, dans les limites imposées par l'ordonnancement considéré, puis vers la gauche, dans le but d'obtenir un ordonnancement améliorant.

### 2.2.3 Les algorithmes évolutionnistes

#### Algorithmes génétiques

Les algorithmes génétiques requièrent une *représentation* particulière des problèmes sous forme de *gènes*, sur lesquels il est possible d'appliquer les mécanismes de l'*évolution*. Dans le cadre du RCPSP, de nombreuses méthodes ont pris le parti de s'appuyer sur la représentation par *liste d'activités*. Une telle liste $L = \{l_1, \ldots, l_n\}$ est réalisable en regard des contraintes de précédence : une activité n'apparaît dans la liste qu'après tous ses prédécesseurs. La liste est couplée à une procédure de *décodage* permettant de générer l'ordonnancement qui y est associé. Cette procédure consiste généralement à effectuer un simple appel au SGS sériel, la règle de priorité étant remplacée par la liste. Ainsi, la construction d'un ordonnancement est réalisée en traitant les activités dans l'ordre de la liste et en les ordonnançant à la plus petite date permise par les contraintes de précédence et de ressources.

Dans son algorithme génétique auto-adaptatif, Hartmann [Hartmann 2002] travaille sur une représentation sensiblement différente. La population comporte en effet des individus composés chacun d'une liste d'activités couplée à un gène auxiliaire dont la valeur détermine le SGS utilisé lors du décodage de la liste : 1 pour le SGS sériel, 0 pour le parallèle. Alors que la plupart des algorithmes génétiques utilisent le SGS sériel à l'exclusion de toute autre méthode de décodage, l'intérêt de l'approche réside précisément dans la possibilité d'employer alternativement le SGS sériel ou le SGS parallèle. En effet, relativement à l'instance considérée, un des schémas peut prendre le pas en terme de qualité des solutions sur l'autre : les projets comportant beaucoup d'activités et/ou dont les capacités de ressources sont ténues semblent ainsi être mieux résolus par le SGS parallèle ([Kolisch 1996], [Hartmann 2000]). L'idée est alors de favoriser par le biais des règles de *croisement* et de *sélection* le schéma le plus efficace sur l'instance considérée. Ainsi, les configurations menant aux meilleures solutions seront les plus à même de survivre au cours des générations successives, et avec elles le gène codant pour la procédure de décodage la plus efficace.

La population, initialement générée à l'aide d'une heuristique de type échantillonnage partiellement aléatoire, est confrontée à l'application des opérateurs usuels d'évolution au cours d'itérations successives. La filiation est réalisée par le biais d'un opérateur de croisement deux points classique auquel s'ajoute un héritage spécifique du gène codant pour le SGS utilisé : la mère transmet son gène à la fille, le père au fils. Chaque nouvel individu produit par croisement est soumis à un opérateur de mutation qui agit à la fois sur la liste et le gène auxiliaire : ce dernier aura une probabilité $p = 0,05$ de prendre la valeur complémentaire. La règle de sélection utilisée pour déterminer la génération suivante consiste à conserver les meilleurs individus parmi les parents et les enfants nouvellement produits.

Une procédure similaire a été mise au point par Alcaraz et Maroto [Alcaraz 2001]. L'algorithme travaille sur des individus constitués d'une liste d'activité alliée à un gène supplémentaire indiquant la procédure de décodage utilisée. Cependant, le choix ne porte pas ici sur l'emploi du schéma sériel ou parallèle mais sur le sens d'ordonnancement : suivant la valeur du gène, la solution est construite au cours d'une passe avant ou arrière du SGS sériel. De la même façon que précédemment, la descendance est générée à l'aide d'un opérateur de croisement deux points, adapté dans ce cas à la gestion du forward-backward, la fille héritant du gène déterminant la procédure de décodage de

sa mère, le fils de son père.

D'autres algorithmes utilisent un encodage différent des solutions : *clé aléatoire* (Gonçalves et Mendes [Gonçalves 2003]), *vecteur de règles de priorité* (Hartmann [Hartmann 1998]). Une approche sporadique développée par Toklu [Toklu 2002] travaille même directement sur des *vecteur de dates de début* des activités. Une fonction de pénalité évaluant les conflits de ressources est utilisée pour pallier le fait que les opérateurs de croisement n'assurent pas systématiquement la production d'ordonnancements réalisables à partir de deux individus.

#### Colonies de fourmis

Merkle et al. [Merkle 2002] présentent la première application d'une procédure d'optimisation par colonies de fourmis. A chaque génération, $m$ fourmis sont produites, correspondant chacune à une application du SGS sériel. La sélection à l'ordonnancement d'une activité candidate est fonction de la combinaison d'une règle de priorité et d'une information *phéromonale*. Cette information est le produit de l'*apprentissage* réalisé par les fourmis au cours des générations successives : la quantité de phéromone associée au séquencement d'une activité $i$ en position $j$ varie de concert avec la qualité des ordonnancements obtenus lorsque ce séquencement est effectif. Des procédés d'*évaporation* de phéromone et d'oubli de la meilleure solution connue sont mis en oeuvre afin d'éviter la convergence trop rapide de l'algorithme. L'application d'une procédure de recherche locale (voisinage 2-opt) à la meilleure solution d'une génération permet d'améliorer les résultats.

### 2.2.4 Les algorithmes de recherche locale

#### Voisinage

Le principe des méthodes de recherche locale consiste à explorer à chaque itération le voisinage d'une solution courante et à en extraire une solution. Le point-clé de ces approches réside donc dans la définition qu'elles donnent du voisinage d'une solution. Dans le cadre du RCPSP, ou de problèmes connexes (tel le job-shop), de nombreux voisinages ont été imaginés. Ceux-ci peuvent résulter d'opérations sur la représentation du problème ou de considérations relatives à l'ordonnancement associé à la solution courante.

Les voisinages élaborés sur la base d'une modification de la liste d'activités peuvent par exemple faire intervenir la *permutation* de deux activités (generalised pairwise interchange : [Della Croce 1995]) ou le *décalage* d'une activité (shift operator). Dans tous les cas, la liste résultante doit demeurer réalisable vis-à-vis des contraintes de précédence. Des opérateurs de décalage plus sophistiqués ont été proposés par Baar et al. [Baar 1998]. Le but est de supprimer un arc *critique* dans le graphe $G = (V, A^s), A^s = \{(i, j), s_i + p_i = s_j\}$, condition essentielle à l'amélioration de la solution. Pour ce faire, ils étendent le procédé de décalage simple en permettant à plusieurs activités d'être déplacées simultanément. D'autres opérateurs prennent en considération la destruction de chemins critiques pour constituer leur voisinage, notamment dans le cadre de problèmes de job-shop : échange de deux activités séquencées sur la même machine permettant de supprimer un chemin critique, conservation des ordres d'exécution des activités non-critiques sur $k$ (parmi $m$) machines ([Applegate 1991]), ...

Dans un ordre d'idées différent, des voisinages peuvent être générés en considérant des *blocs* d'activités possédant une propriété particulière : exécution à l'intérieur d'une fenêtre de temps donnée, ordre d'exécution déterminé sur chaque machine (job-shop). Il peut alors s'agir de réordonnancer un bloc indépendamment des autres [Mausser 1997], ou bien de réagencer l'exécution des activités hors-bloc ([Caseau 1999a]), ... Un autre type de voisinage porte sur la conservation partielle, aléatoire ou non, des informations de séquencement relatifs des activités ([Applegate 1991], [Baptiste 1995]). Les voisins sont alors constitués par les ordonnancements réalisables satisfaisant ces contraintes de séquencement.

Les voisinages considérés dans les paragraphes suivants porteront exclusivement sur des opérations réalisées sur des listes d'activités, à l'exception de la méthode de Artigues et al. [Artigues 2000]. En effet, toutes les autres méthodes décrites ci-après utilisent ce mode de représentation, la procédure de décodage associée étant le SGS sériel.

### Recuit simulé

Slowinski et al. [Slowinski 1994] furent les premiers à s'atteler à la résolution du RCPSP, dans sa version multi-mode, à l'aide d'une procédure de recuit simulé. Le voisinage préconisé est basé sur l'échange de deux activités choisies aléatoirement, telles que celles-ci ne possèdent pas de relation de précédence. Boctor [Boctor 1996], et plus récemment Bouleimen et Lecocq [Bouleimen 2003], se sont par la suite essayés à l'exercice. La différence avec l'approche précédente réside dans l'opérateur déterminant le voisinage (opérateur de décalage).

### Recherche taboue

Dans le cadre des méthodes taboues, la différence majeure entre les différentes approches réside dans la définition du voisinage. Nonobe et Ibaraki [Nonobe 2002] suggèrent l'utilisation de l'opérateur de décalage associé à un mécanisme de réduction de voisinage spécifique permettant une énumération plus rapide de ce dernier. Pinson et al. [Pinson 1994] proposent différentes variantes d'une même procédure. Les alternatives portent sur la façon dont est généré le voisinage : adjacent pairwise interchange, (general) pairwise interchange, opérateur de décalage. Baar et al. [Baar 1998] ont développé deux algorithmes distincts. Le premier fait intervenir des opérateurs de décalage, influant sur le chemin critique. La seconde approche, moins classique, repose sur une représentation par schéma d'ordonnancement (introduit par Brucker et al. [Brucker 1998]) associée à une procédure de décodage et une définition du voisinage appropriée. Dans les deux cas, la liste taboue est gérée de façon dynamique et la solution initiale est générée à l'aide d'une heuristique à règle de priorité.

Une approche plus originale est abordée par Artigues et al. [Artigues 2000]. Celle-ci étend la notion de graphe *disjonctif* au RCPSP. Les solutions sont ainsi représentées par un *multi-flot* et le voisinage est défini comme l'ensemble des réinsertions des tâches critiques dans le graphe des flots.

### 2.2.5 Les algorithmes hybrides

Le champ des méthodes basées sur la coopération de divers composants heuristiques s'agrandit de jour en jour. D'une conception souvent très sophistiquée, elles se révèlent néanmoins très performantes au niveau de la qualité des résultats obtenus.

Valls et al. [Valls 2002] incorporent à un algorithme génétique classique basé sur la représentation des solutions par liste d'activités leur procédure d'amélioration de forward-backward [Valls 2004a]. Celle-ci est appliquée à chaque individu généré au cours du processus évolutionniste. La technique de croisement invoquée fait intervenir la notion de *pic* dans l'utilisation des ressources : un pic dans une liste d'activités correspond à une portion de cette liste telle que la consommation de ressources par les activités qui la constituent, lorsqu'elles sont ordonnancées, est supérieure à une certaine proportion de ressources disponibles. Les individus produits par un tel croisement héritent le pic d'un de leurs parents, le reste de la liste étant complété par l'autre parent. Outre cet opérateur de croisement singulier, une autre originalité de la méthode est qu'elle initie une seconde phase d'évolution à partir d'une population constituée de voisins de la meilleure solution actuelle. Ceux-ci sont construits à l'aide de la procédure utilisée par Valls et al. [Valls 2000] appliquée ici à la représentation par liste d'activités.

Kochetov et Stolyar [Kochetov 2003a] proposent quant à eux un algorithme évolutionniste combinant algorithme génétique, path relinking et recherche taboue. Les solutions se développent et se diversifient de façon génétique, tandis que le processus d'évolution est réalisé en construisant le chemin contenant les solutions menant aux deux solutions sélectionnées. La meilleure solution du chemin est ensuite soumise à l'application d'une procédure taboue. Celle-ci emploie un voisinage qui découpe la liste d'activités en trois parties distinctes. Pour les parties situées aux extrémités de la liste, le schéma sériel est utilisé tandis que la partie centrale est décodée par la procédure parallèle. La meilleure solution retournée par la recherche taboue est ajoutée à la population, remplaçant la solution de qualité la plus médiocre.

En marge de ces hybridations à présent usuelles, des méthodes faisant collaborer des procédés de résolution locale et heuristique voient le jour, ainsi que nous avons pu le constater au cours du chapitre précédent. Cependant, dans le cadre particulier du RCPSP, l'approche demeure singulière, malgré le succès rencontré par des méthodes analogues sur d'autres problèmes d'ordonnancement (job-shop notamment) : Forget & Extend [Caseau 1999a], Shifting Bottleneck [Adams 1988]. La méthode de Sprecher [Sprecher 2002] tente toutefois d'aller dans ce sens. L'approche retenue consiste à décomposer le projet global en sous-projets distincts puis à les ordonnancer à l'aide de la procédure arborescente décrite dans [Sprecher 1996] avant de concaténer les solutions afin de reconstituer la solution globale. La décomposition est basée sur le découpage d'une séquence d'activités, satisfaisant à la fois les contraintes de précédence et une propriété de monotonie des dates de début, le nombre et la taille des sous-problèmes étant des paramètres de l'algorithme. On peut constater que cette approche, si elle présente des résultats compétitifs sur les instances de taille modérée, s'écroule sur celles de taille importante, tant au niveau de la qualités des solutions que des temps de calcul dispensés. Elle arrive de plus difficilement à établir un compromis : une décomposition en de nombreux sous-problèmes (de taille modérée donc) permet de réduire les temps d'exécution mais au détriment notable de l'aspect qualitatif de l'approche, tandis qu'une décomposition en peu

de sous-problèmes fait exploser les temps de calcul, sans pour autant améliorer de façon vraiment significative les résultats.

## 2.3 Une première application de LSSPER au RCPSP

Le chapitre précédent présentait le schéma général d'exécution d'une méthode de grand voisinage. Cette section est à présent consacrée à son application au problème particulier de RCPSP.

Contrastant avec la plupart des heuristiques de la littérature, aucune représentation particulière des solutions, telle les listes d'activités, vecteurs de priorités ou autres graphes de disjonction, n'est ici utilisée. Ainsi, une solution est donnée par le vecteur $\overline{S}$ de dates de début des activités, réalisable en regard des contraintes de précédence et de ressources.

Nous débutons cet exposé par la présentation de l'algorithme en section 2.3.1. Ce dernier reprend celui donné au chapitre précédent en spécifiant les techniques particulières mises en oeuvre lors des différentes phases clé de la méthode, que nous développons par la suite :

- en section 2.3.2, nous présentons une méthode de forward-backward (FB) qui constitue un rouage important du processus global.
- nous envisageons en sections 2.3.3 et 2.3.4 les moyens de génération et résolution de sous-problèmes.
- nous abordons les techniques de post-optimisation et de diversification de la recherche mises en oeuvre en sections 2.3.5 et 2.3.6.

Nous clôturerons finalement cette étude par la présentation de résultats expérimentaux obtenus sur des jeux d'instances classiques en section 2.3.7.

### 2.3.1 Algorithme général

L'algorithme, schématisé en Figure 2.2, débute par la construction d'une solution initiale $\overline{S}^0$. Le processus itératif de génération et résolution de sous-problèmes est alors initié. Il consiste en plusieurs phases de descente entremêlées d'un processus de diversification de la recherche.

A chaque itération $s \geq 1$, un sous-problème $\Pi^s$ de taille $p$ auto-adaptative, initialisée à une borne inférieure $\underline{p}$ est généré à partir de la solution courante $\overline{S}^{s-1}$. Ce sous-problème, de type RCPSP avec fenêtres de temps et capacités variables de ressources, est ensuite résolu par une méthode de recherche arborescente tronquée. Si au bout d'un temps de calcul limité à une borne supérieure $H$ une solution à $\Pi^s$ est trouvée, celle-ci est post-optimisée afin d'obtenir la solution voisine $\overline{S}^s$. Sinon la solution courante $\overline{S}^{s-1}$ est conservée. Enfin, un processus de diversification de la recherche permet de réinitialiser périodiquement la solution courante.

Le processus global s'achève lorsqu'un nombre maximal d'itérations est atteint ou lorsque la valeur de la meilleure solution trouvée $\overline{S}^*$ coïncide avec une borne inférieure LB (borne de Zaloom [Zaloom 1971]).

**Début**
1. Génération de la solution initiale $\overline{S}^0$ par une méthode de forward-backward
2. $\overline{S}^* := \overline{S}^0$, $s := 1, p = \underline{p}$
3. **Répéter**
4. Génération du sous-problème $\Pi^s$ de taille $p$ à partir de $\overline{S}^{s-1}$ (RCPSP-TW-VC)
5. Résolution de $\Pi^s$ à l'aide d'une méthode exacte tronquée (limite de temps $H$)
6. **Si** une solution à $\Pi^s$ est obtenue **Alors**
7. Obtention de la solution voisine $\overline{S}^s$ par post-optimisation
8. **Sinon**
9. $\overline{S}^s := \overline{S}^{s-1}$
10. **Si** $f(\overline{S}^s) < f(\overline{S}^*)$ **Alors**
11. $\overline{S}^* := \overline{S}^s$
12. Processus de diversification de la recherche
13. $s := s + 1$
14. **Jusqu'à** $s := MAX\_ITER$ ou $f(\overline{S}^*) = LB$

**Fin**

FIG. 2.2 – *Algorithme général de LSSPER pour le RCPSP*

### 2.3.2 Heuristique de forward-backward et règles de priorité

En section 2.2.2, nous avons décrit le principe des heuristiques de forward-backward qui construisent des ordonnancements successifs au cours de passes avant et arrière alternées. Nous allons à présent présenter une heuristique basée sur le même schéma mais possédant des spécificités assurant sa convergence. Celle-ci prend en charge la réalisation de diverses fonctionnalités de la méthode : génération de la solution initiale, diversification de la recherche et post-optimisation.

L'approche consiste à produire de façon alternative des ordonnancements au plus tôt et au plus tard, générés respectivement au cours de passes avant et arrière faisant appel au SGS sériel. La particularité de cette heuristique par rapport à celles présentées en section 2.2.2 réside dans les règles de priorité employées au cours des passes successives :

- INCR_BST : appliquée lors des passes avant, à l'exception de la première. Les activités sont classées dans l'ordre des dates de début croissantes obtenues au cours de la passe arrière précédente.
- DECR_FCT : appliquée lors des passes arrière. Les activités sont classées dans l'ordre des dates de fin décroissantes obtenues au cours de la passe avant précédente.

L'algorithme s'achève lorsque les valeurs de makespan trouvées à l'issue de deux passes successives coïncident.

Le couplage de ce critère d'arrêt avec les règles de priorité précédemment énoncées assure la convergence de l'algorithme : d'une part, les ordonnancements successifs ne peuvent pas dégrader la valeur de makespan, d'autre part le critère d'arrêt impose une amélioration continuelle de cette valeur au cours des passes successives, sans quoi le processus s'achève.

**Proposition 1** *Soit* $\overline{S} = (\overline{s}_0, \ldots, \overline{s}_{n+1})$*, un ordonnancement réalisable et* $\overline{S}' = (\overline{s}'_0, \ldots, \overline{s}'_{n+1})$ *tel*

*que :*

$$\overline{s}'_i \leq \overline{s}_i \ \forall i \in V \tag{2.4}$$

*2.4 est une condition suffisante pour que* $\overline{S}'$ *ne dégrade pas* $\overline{S}$*.*

**Preuve.** De façon triviale, 2.4 est vraie car $\overline{s}'_{n+1} \leq \overline{s}_{n+1}$.

**Proposition 2** *Soit* $\overline{S}^B$*, le vecteur des dates de début obtenu après une passe arrière. La passe avant lui succédant génère un ordonnancement* $\overline{S}^F$ *qui respecte 2.4, i.e. qui ne dégrade pas* $\overline{S}^B$*.*

**Preuve.** Considérons que les tâches sont numérotées dans l'ordre croissant de leurs dates de début dans $\overline{S}^B$. Montrons par récurrence que $\overline{s}_i^F \leq \overline{s}_i^B \ \forall i = 0, \ldots, n+1$.

La proposition est vraie pour $i = 0$ car $\overline{s}_0^F = \overline{s}_0^B = 0$.

Supposons qu'elle soit vraie pour $j = 0, \ldots, i$ et démontrons qu'elle reste vraie pour $j = i + 1$.

L'ordonnancement partiel $\{\overline{s}_1^F, \ldots, \overline{s}_i^F, \overline{s}_{i+1}^B\}$ est réalisable car $\overline{s}_j^F \leq \overline{s}_j^B \ \forall j = 0, \ldots, i$. Comme l'algorithme sériel cale les tâches au plus tôt, on a $\overline{s}_{i+1}^F \leq \overline{s}_{i+1}^B$.

D'où 2.4 est vraie pour $\overline{S}^F$ et $\overline{s}_{n+1}^F \leq \overline{s}_{n+1}^B$.

Le raisonnement permettant de prouver qu'une passe arrière ne dégrade pas le résultat retourné par la passe avant précédente est similaire.

**Théorème 3** *L'heuristique proposée converge en au plus UB-LB+1 pas.*

**Preuve.** Nous avons démontré que la valeur de la durée totale du projet ne peut pas, d'une passe à l'autre, être dégradée. De même celle-ci ne peut rester constante au cours de deux passes successives, auquel cas le test d'arrêt de l'heuristique est satisfait. Si l'on borne la valeur du projet supérieurement et inférieurement par les valeurs UB et LB, alors on peut conclure que l'heuristique converge dans le pire des cas en UB-LB+1 itérations.

### Ajustements pour les ordonnancements au plus tard

Au cours d'une passe arrière, l'ordonnancement est construit à partir de l'instant de référence correspondant à la valeur de makespan trouvé lors de la passe avant précédente. En effet, la propriété de non-dégradation de la valeur de makespan garantit l'obtention d'un ordonnancement réalisable ne possédant pas d'activités s'exécutant dans le passé.

Pour la même raison, il arrive parfois que la date de début du projet $\overline{s}_0^B$ ne coïncide pas avec l'instant de référence $t = 0$. La solution est alors obtenue en décalant vers la gauche de $\overline{s}_0^B$ unités de temps l'exécution de toutes les activités : $\overline{s}_i^B = \overline{s}_i^B - \overline{s}_0^B \ \forall i \in V$.

### Règles de priorité supplémentaires

L'heuristique assure diverses fonctions au sein du processus global. A chacune de ces fonctionnalités est associée une règle spécifique de priorité, utilisée lors de la première passe avant :

- MINLFT : à chaque itération, l'activité candidate sélectionnée est celle possédant la plus petite date de fin au plus tard [1].

[1] pour chaque activité la date de fin au plus tard est donnée par la différence entre une borne supérieure et la valeur du plus long chemin entre le sommet correspondant dans le graphe potentiels-tâches et le sommet fictif de fin

– RANDOM : à chaque itération, l'activité à ordonnancer est choisie de façon aléatoire parmi l'ensemble des activités candidates.
– INCR_STC : les activités sont ordonnancées dans l'ordre des dates de début croissantes de la solution courante.

### 2.3.3 Génération auto-adaptative d'un sous-problème

A chaque itération $s \geq 1$ de la méthode, un sous-problème $\Pi^s$, et par voie de conséquence un voisinage $V(s)$, sont définis par la sélection d'un ensemble $A^s \subseteq A$ de $p$ activités $A^s = \{i_1, \ldots, i_p\}$, à l'exclusion des activités fictives 0 et $n+1$ : toute activité $j \in A \setminus A^s$ étant "gelée" à la date de début $\overline{S}_j^{s-1}$ qu'elle occupe dans la solution courante, il s'agit alors de trouver une solution réalisable pour le sous-projet défini par les activités restantes. Nous décrivons ci-après les contraintes pour l'obtention d'une solution réalisable de $\pi^s$ et donc d'une solution voisine de $S^{s-1}$. Nous présenterons ensuite la fonction objectif utilisée lors de la recherche de cette solution.

En pratique, geler la date de début de certaines activités est équivalent à générer un problème d'ordonnancement avec fenêtres de temps et capacités variables des ressources :

– le problème à ordonnancer peut être modélisé par un graphe potentiels-tâches réduit :

$$G^s = (A^s, E^s), E^s = \{(i,j) | i,j \in A^s, (i,j) \in E\}$$

– les capacités des ressources sont modifiées afin de prendre en considération les périodes d'indisponibilité partielle induites par l'exécution des tâches gelées : pour chaque ressource $k \in \mathcal{R}$, la quantité $R_{k\tau}$ définit la capacité de celle-ci à l'instant $\tau$ :

$$R_{k\tau} = R_k - \sum_{j \in (A \setminus A^s) \cap \mathcal{A}(\tau, \overline{S}^{s-1})} r_{jk}$$

– chaque activité $i \in A^s$ est associée à une fenêtre de temps $[ES_i, LS_i]$ calculée en fonction de ses prédécesseurs et successeurs mais également des disponibilités de ressources. Le processus de détermination des dates de début au plus tôt des différentes activités est donné en Figure 2.3.

---

**Début**
1. **Initialisation** : $ES_i = -1 \ \forall i \in A^s$
2. **Tant que** $\exists i, ES_i = -1$
3. Choisir une activité $i$ telle que $ES_i = -1$ et $\Gamma_i^{-1} = \emptyset$ ou $ES_j \neq -1, \forall j | (j,i) \in E^s$
4. $ES_i^{pred} = \max\{\max_{j \in A \setminus A^s, (j,i) \in E} \overline{S}_j^{s-1} + p_j, \max_{j \in A^s, (j,i) \in E^s} ES_j + p_j\}$
5. $ES_i = \min\{t | t \geq ES_i^{pred}, \forall k \in R, \forall \tau = t, \ldots, t + p_i - 1, R_{k\tau} \geq r_{ik}\}$

**Fin**

---

FIG. 2.3 – *Détermination des dates de début au plus tôt*

A chaque itération, une activité $i$ ne possédant pas de prédécesseur non encore traité (pas 3) se voit d'abord associer une date $ES_i^{pred}$ équivalente à la date de début la plus petite permise par

les contraintes de précédence avec les activités gelées et sélectionnées (pas 4). Un ajustement est ensuite réalisé en tenant compte des disponibilités de ressources : l'activité $i$ doit pouvoir s'exécuter sans préemption à partir de la date $ES_i$ (pas 5).

Le calcul des dates de début au plus tard de chacune des activités s'effectue de façon symétrique. Le processus est donné en Figure 2.4. Comme il ressort du calcul de $LS_i^{succ}$, chaque activité est contrainte d'être terminée au makespan courant $\overline{S}_{n+1}^{s-1}$.

---

**Début**
1. **Initialisation** : $LS_i = \infty \; \forall i \in A^s$
2. **Tant que** $\exists i, LS_i = \infty$
3. Choisir une activité $i$ telle que $LS_i = \infty$ et $\Gamma_i = \emptyset$ ou $LS_j \neq \infty, \forall j | (i,j) \in E^s$
4. $LS_i^{succ} = \min\{\min_{j \in A \setminus A^s, (i,j) \in E} \overline{S}_j^{s-1} - p_i, \min_{j \in A^s, (i,j) \in E^s} LS_j - p_i\}$
5. $LS_i = \max\{t | t \leq LS_i^{succ}, \forall k \in R, \forall \tau = t, \ldots, t + p_i - 1, R_{k\tau} \geq r_{ik}\}$

**Fin**

---

FIG. 2.4 – *Détermination des dates de début au plus tard*

Le sous-problème étant à présent défini, il s'agit alors de déterminer les dates de début $(S_{i_1}^s, \ldots, S_{i_p}^s)$ des activités de façon à obtenir le "meilleur" voisin possible. Notre choix s'est dans un premier temps porté sur la minimisation de la durée totale $\tilde{C}_{max}$ du sous-projet. Nous reviendrons sur cet objectif dans la section 2.4 pour proposer d'autres critères associés à de nouvelles stratégies de sélection du sous-problème.

Conformément aux contraintes et objectif définis précédemment, le sous-problème peut se formuler de la façon suivante :

$$(\Pi^s) \qquad \min \tilde{C}_{max} \tag{2.5}$$

$$\tilde{C}_{max} \geq S_i^s + p_i \qquad \forall i \in A^s \tag{2.6}$$

$$S_i^s \geq ES_i \qquad \forall i \in A^s \tag{2.7}$$

$$S_i^s \leq LS_i \qquad \forall i \in A^s \tag{2.8}$$

$$S_j^s \geq S_i^s + p_i \qquad \forall (i,j) \in E^s \tag{2.9}$$

$$\sum_{i \in A^s \cap \mathcal{A}(\tau, S^s)} r_{ik} \leq R_{k\tau} \quad \forall k \in R, \forall \tau \in \{0, \ldots, \overline{S}_{n+1}^{s-1}\} \tag{2.10}$$

La Figure 2.5 illustre le processus de génération d'un sous-problème. A partir de la solution donnée en partie (b), et considérant que la valeur de $p$ vaut 4 et que les activités $A^s = \{7, 8, 9, 10\}$ ont été sélectionnées, nous pouvons définir le sous-problème de la façon suivante : il s'agit d'ordonnancer le projet modélisé par le graphe potentiels-tâches donné en partie (a), de telle façon que les activités s'exécutent pendant les fenêtres de temps définies en partie (c) et qu'à tout instant la consommation, par les activités en cours d'exécution, de la ressource $r_1$, dont le profil est représenté en partie (d), n'excède pas sa capacité, variable dans le temps.

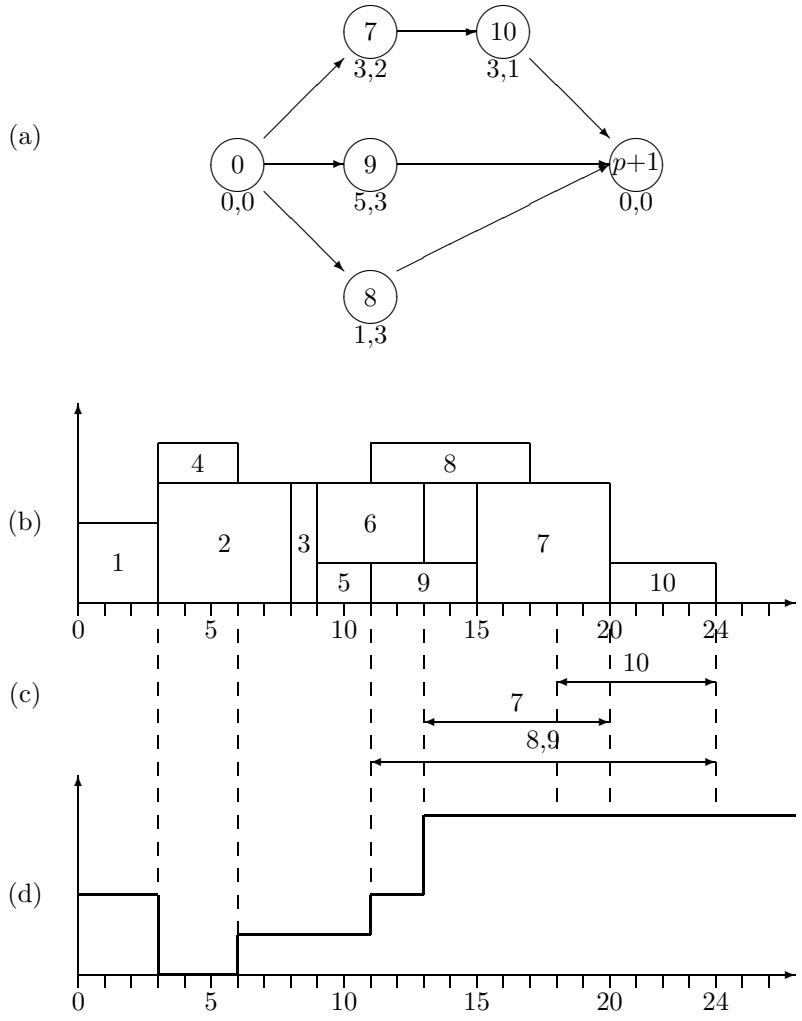


FIG. 2.5 – *Exemple de génération de sous-problème pour* $p = 4$

La construction de l'ensemble $A^s$ a été envisagée selon différentes méthodes de sélection des $p$ activités :

- Aléatoire&Critique : les activités sont sélectionnées de façon aléatoire mais une probabilité plus élevée est assignée aux activités critiques. La caractère de criticité d'une activité est déterminé par des dates de début identiques dans les ordonnancements au plus tôt et au plus tard obtenus au terme des deux dernières passes de l'heuristique de forward-backward.
- Aléatoire&Prédécesseurs_Projet. Les activités sont sélectionnées de façon aléatoire mais lorsqu'une activité $i$ est sélectionnée, tous ses prédécesseurs immédiats dans le projet le sont également, c.-à-d. toute activité $j$ telle que $(j, i) \in E$.
- Aléatoire&Prédécesseurs_Ordo. Les activités sont sélectionnées de façon aléatoire mais lorsqu'une activité $i$ est sélectionnée, tous ses prédécesseurs immédiats dans l'ordonnancement

le sont également, c.-à-d. toute activité $j$ vérifiant $\overline{S}_j^{s-1} + p_j = \overline{S}_i^{s-1}$.

- Aléatoire&Tous_Prédécesseurs. Les activités sont sélectionnées en combinant les deux stratégies précédentes.
- Bloc. A partir d'une première activité $i$ sélectionnée, toute activité $j$ dont l'exécution est contiguë ou concomitante à celle de $i$, c.-à-d. telle que $\overline{S}_i^{s-1} - p_j \leq \overline{S}_j^{s-1} \leq \overline{S}_i^{s-1} + p_i$, est ajoutée dans l'ensemble $A^s$, dans la limite de $p$ activités. Si cette limite n'est pas atteinte, alors le processus est itéré à partir de la deuxième, puis troisième, ..., activité ajoutée à l'ensemble $A^s$. Ce schéma de sélection peut être considéré comme une variation des stratégies employées dans [Caseau 1999a] et [Mausser 1997]. Le sous-problème représenté en Figure 2.5 est obtenu par ce schéma de génération à partir de l'activité 7 : les activités 8, 9 et 10 vérifiant la condition de simultanéité ou de contiguïté d'exécution sont incluses dans le sous-problème.

En ce qui concerne l'ensemble des stratégies, à l'exception de la première, le but est de générer un sous-problème suffisamment indépendant du reste de la solution afin d'obtenir un voisinage qui permette une exploration significative de l'espace de recherche. Les stratégies successives élaborées vont d'ailleurs dans le sens d'une indépendance croissante du sous-problème par rapport aux activités qui demeurent fixées, indépendance qui varie de concert avec l'efficacité de la méthode globale (cf. section 2.3.7).

De la même façon, le voisinage considéré se doit d'être suffisamment grand, afin d'offrir la possibilité de s'extraire des optima locaux. On cherche ainsi à maximiser sa taille tout en s'assurant que celle-ci demeure raisonnable, c.-à-d. de telle façon que la résolution des sous-problèmes puisse être réalisée efficacement. Ainsi, la taille $p$ des sous-problèmes, influant de façon directe sur la taille de l'espace de recherche, est-elle fonction des temps moyens de résolution des sous-problèmes successifs. La variation de $p$ au cours des itérations est fermement liée à celle du paramètre $t = \sum_{q=s-1,\ldots,s-5} t^q$, qui représente le temps cumulé de résolution des sous-problèmes sur les cinq dernières itérations. Les règles empiriques d'auto-ajustement de $p$ à la fin d'une itération $s$ en fonction de la valeur de $t$ sont les suivantes :

- si $t \leq H$, alors $p = p + 2$ ;
- si $H < t \leq 5H/2$, alors $p = p + 1$ ;
- si $5H/2 < t \leq 4H$, alors $p = p - 1$ ;
- si $4H < t \leq 5H$, alors $p = p - 2$.

Aucune diminution de la valeur de $p$ n'est réalisée si ce dernier atteint une limite inférieure $\underline{p}$ : le voisinage considéré est assuré de demeurer de taille "raisonnable". De la même façon, aucune augmentation n'est admise au-delà d'une limite supérieure $\overline{p}$.

### 2.3.4 Résolution d'un sous-problème

Notre approche consiste à évaluer la généralité de la méthode proposée ainsi que son applicabilité aux situations pratiques en délégant la résolution du sous-problème à un solveur commercial. Les expériences menées ont fait intervenir l'utilisation de deux solveurs distincts, afin de déterminer de façon probante l'influence ou non de la méthode de résolution choisie sur les performances globales : un solveur de programmation par contraintes (CP), dédié à la résolution de problèmes d'ordonnancement, et un solveur de programmation linéaire en nombres entiers (ILP) ont ainsi été

éprouvés. Dans les deux cas, la méthode de résolution consiste en une recherche arborescente pour laquelle aucun paramétrage particulier n'a été réalisé (cf. section 2.3.7). Le solveur retourne, le cas échéant, la meilleure solution trouvée au bout de la limite de temps $H$ qui lui a été fixée.

Du fait de l'existence d'une librairie spécialisée, l'écriture des contraintes est directe dans le cas du solveur CP, alors que le solveur ILP nécessite l'écriture d'un modèle qui constitue une extension au cas de ressources variables et fenêtres de temps de la formulation donnée par Pritsker *et al.* [Pritsker 1969]. De fait, ce modèle est particulièrement adapté au cas de ressources de capacités non-constantes car il intègre, pour chaque ressource, des contraintes portant sur une disponibilité par période de cette dernière.

En partie (a) de la Figure 2.6, la solution retournée par le solveur pour le sous-problème de la Figure 2.5 est représentée. La solution $\overline{S}^s$ obtenue par extension directe de cette solution est figurée en partie (b). Dans ce cas, une solution améliorante (et optimale) de makespan 22 est obtenue.

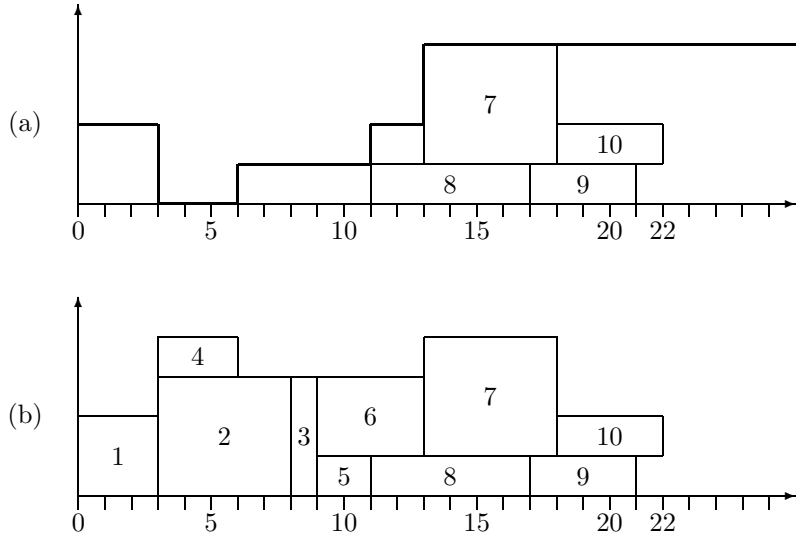


FIG. 2.6 – *Résolution du sous-problème à 4 activité de la Figure 2.5*

### 2.3.5 Post-optimisation

Nous avons vu dans la section précédente qu'une solution réalisable $\overline{S}^s$, voisine de $\overline{S}^{s-1}$, peut être obtenue par extension directe de la solution sous-optimale délivrée par le solveur.

Cependant, dans le cas général, une telle solution peut aisément être améliorée. En effet, toutes les activités ordonnancées à la droite du bloc d'activités constituant le sous-problème sont susceptibles de pouvoir être décalées vers la gauche. Cette post-optimisation est réalisée par l'heuristique de forward-backward : à partir de l'extension directe, et conjointement avec l'utilisation de la règle de priorité DECR_STC, un ordonnancement au plus tôt est généré puis éventuellement amélioré au cours des passes successives.

Ce processus est illustré en Figures 2.7 et 2.8. Une solution de makespan 25 est représentée en partie (a). Les activités 6, 7 et 10 étant fixées à leur valeur courante, un sous-problème constitué des activités restantes est défini puis résolu. Soit la solution sous-optimale, conséquente à la résolution du sous-problème, et son extension directe, représentée en partie (b), nous obtenons trivialement une nouvelle solution globale de makespan 25. Cependant il apparaît de façon évidente que les activités précédemment fixées peuvent être décalées vers la gauche, ce qui est réalisé en exécutant une première passe avant de l'heuristique de forward-backward décrite en 2.3.2, les activités étant ordonnancées dans l'ordre des dates de début croissantes de l'extension directe. Nous obtenons alors l'ordonnancement de makespan 24 figuré en partie (c).

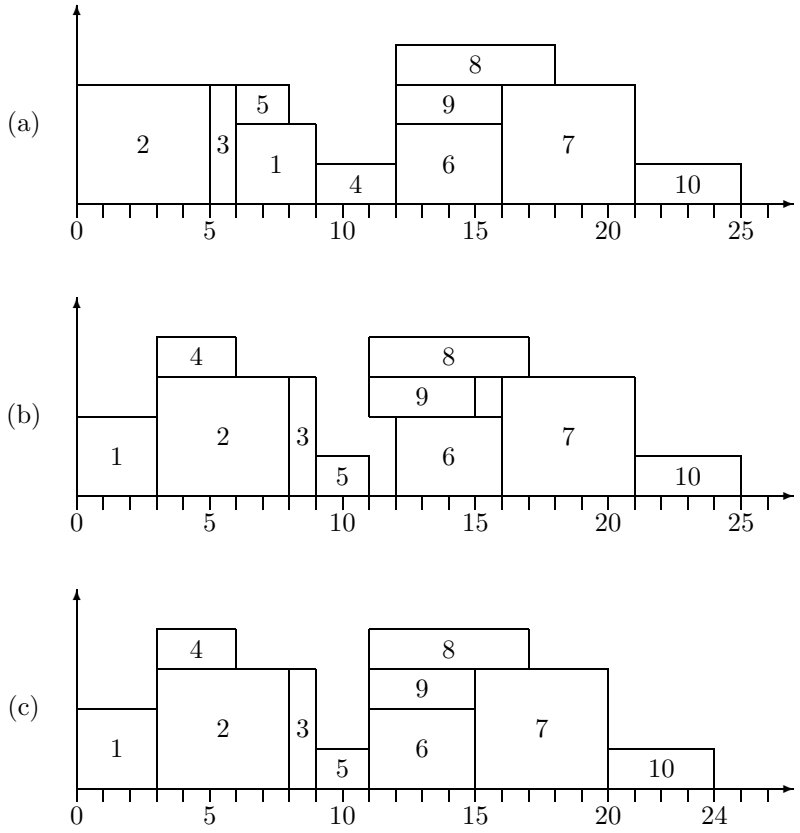


FIG. 2.7 – *Recalage de l'extension directe à l'aide de l'heuristique de forward-backward (I)*

Nous poursuivons ensuite l'exécution de l'heuristique jusqu'à ce que deux passes consécutives génèrent des solutions de makespan équivalent, ainsi que c'est le cas pour les ordonnancements représentés en partie (d) et (e). Nous obtenons à la fin de ce processus la solution voisine, représentée en partie (e), qui est dans ce cas particulier également optimale.

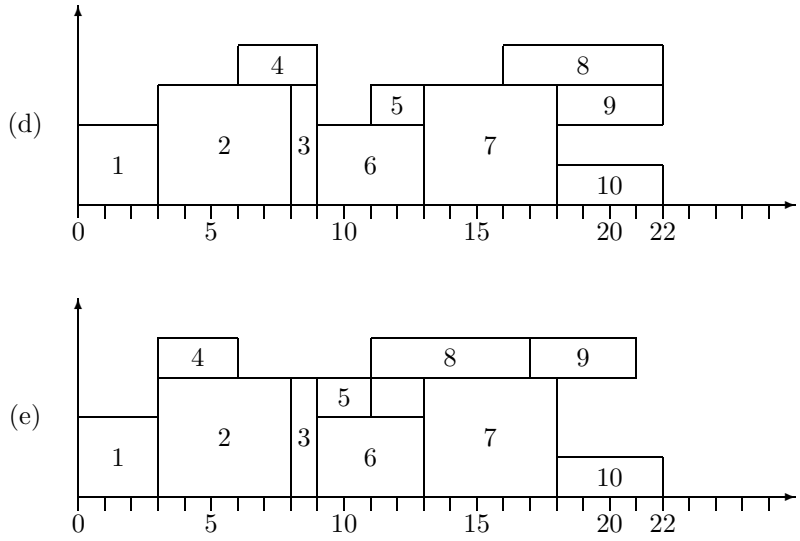


FIG. 2.8 – *Amélioration de l'extension directe à l'aide de l'heuristique de forward-backward (II)*

### 2.3.6 Mécanismes de diversification de la recherche

Nous avons vu en au chapitre précédent que LSSPER propose dans son schéma général des pistes permettant de diversifier la recherche de solutions afin de s'affranchir du piège des minima locaux. Dans le cadre de son application au problème de RCPSP, plusieurs mécanismes ont été mis en place pour assurer cette fonction essentielle.

#### Stratégie de sélection des activités

Une premier point est lié à la génération des sous-problèmes. Non seulement, on cherche à faire en sorte que ces derniers soient le plus indépendants possible du reste de la solution courante afin d'obtenir un voisinage de taille significative, mais en outre une composante aléatoire offre la possibilité d'obtenir des voisinages très différents à partir d'une même solution courante. Ainsi, si l'on considère la méthode de sélection par bloc, les activités sont considérées pour leur inclusion dans $A^s$ dans l'ordre donné par une permutation aléatoire.

La Figure 2.9 illustre ce procédé. A partir de la même solution courante représentée en partie (b), et considérant que la première activité d'inclusion dans $A^s$ est l'activité 6, suivant la place relative des activités 2 et 7 dans la permutation, deux sous-problèmes distincts peuvent être générés puis résolus : en partie (a), la permutation $P_a^s = (\ldots, 2, \ldots, 7, \ldots)$ procure un premier sous-problème $\Pi_a^s$, avec $A_a^s = \{2, 3, 5, 6, 8, 9\}$, dont la résolution n'a aucune incidence sur la solution globale, tandis qu'en partie (c), la permutation $P_c^s = (\ldots, 7, \ldots, 2, \ldots)$ génère un deuxième sous-problème $\Pi_c^s$, distinct de $\Pi_a^s$, avec $A_c^s = \{3, 5, 6, 7, 8, 9\}$, dont la résolution permet une amélioration directe de la valeur globale de makespan (l'activité 10 peut de façon triviale être décalée de deux unités de temps vers la gauche).

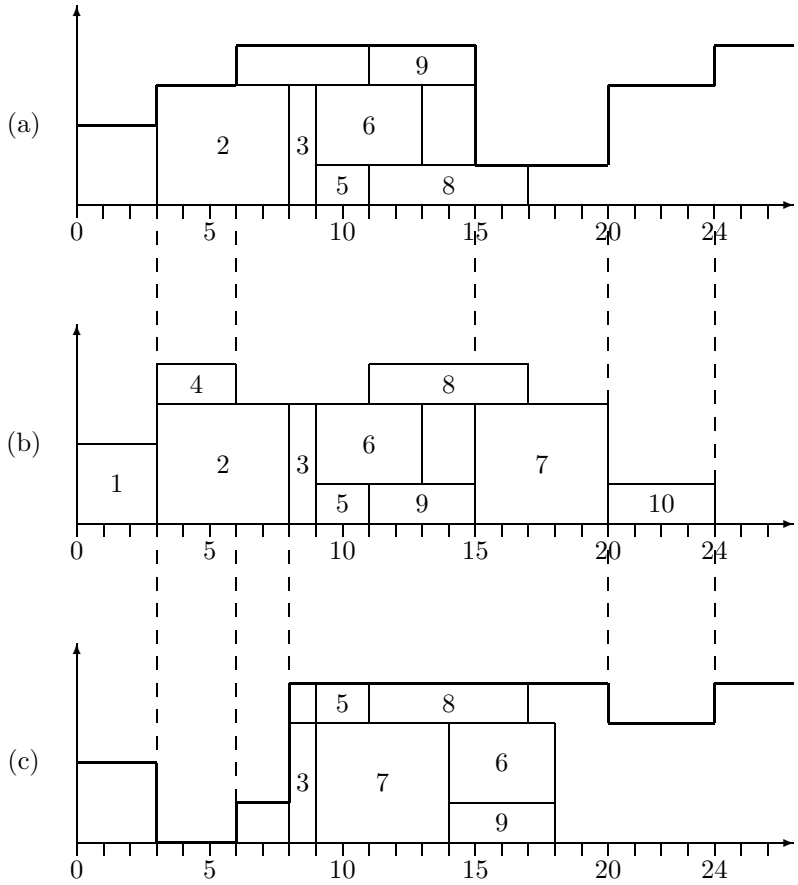


FIG. 2.9 – *Génération et résolution de deux sous-problèmes distincts obtenus à partir d'une même solution courante*

**Reprise de la recherche à partir d'un nouvelle solution**

Périodiquement, lorsque la solution courante n'a pas été améliorée depuis un certain nombre d'itérations, la recherche est stoppée puis redémarrée à partir d'une nouvelle solution. Cette dernière est générée à l'aide de l'heuristique de forward-backward, couplée à l'utilisation de la règle RANDOM lors de la première passe avant qui nous permet de générer une solution possiblement très différente de toutes celles rencontrées jusqu'alors.

### 2.3.7 Résultats expérimentaux

Nous avons implémenté la méthode proposée en C++. Pour la résolution du sous-problème, nous utilisons un solveur de programmation par contraintes spécialisé pour la résolution de problèmes d'ordonnancement : ILOG Scheduler 5.0. Nous avons fixé de façon empirique le temps maximal de résolution $H$ à 0,5 seconde, le nombre total d'itérations à $10 * n$ et la taille initiale (et minimale

admise) des sous-problèmes à 15.

Les expériences ont été menées sur un PC dont le processeur est cadencé à 2.3 GHz et comportant 1 Go de RAM.

La méthode a été éprouvées sur des instances classiques de la littérature :

- Patterson [Patterson 1984] (PAT) : 110 instances à 3 ressources dont la taille varie entre 6 et 51 activités ;
- Baptiste/Le Pape [Baptiste 2000] (BL) : 40 instances à 20 ou 25 activités et 3 ressources ;
- Alverez [Alvarez-Valdés 1989] (ALV) : 48 instances comportant 103 activités et 6 ressources ;
- Kolish *et al.* [Kolisch 1998] (KSD) : 2040 instances à 30, 60, 90 et 120 activités et 4 ressources.

Les instances de Patterson sont à présent réputées faciles à résoudre, à l'inverse des instances KSD pour lesquelles aucune méthode exacte ne parvient à résoudre la totalité des problèmes à 60 activités. De fait, elles sont les plus étudiées de la littérature. Dans une étude de ces instances [Baptiste 2000], Baptiste rapporte que les plus difficiles d'entre elles présentent un fort taux de disjonctions (pourcentage du nombre d'activités ne pouvant deux à deux s'exécuter en parallèle), ce qui est confirmé par d'autres études plus récentes [Garaix 2005]. Baptiste propose des instances possédant au contraire un faible taux de disjonction qu'il désigne comme hautement cumulatives.

Les résultats sont reportés dans le tableau 2.1. Pour chaque jeu d'instances, nous avons renseigné en colonnes 1 et 2 les pourcentages moyens et maximaux de déviation par rapport à la solution optimale (ou à la meilleure solution connue pour les instances ouvertes) et à la borne inférieure du chemin critique. Nous reportons en colonne 3 le nombre de concordances entre la meilleure solution répertoriée à ce jour et le résultat obtenu par LSSPER. Nous indiquons finalement en dernière colonne les temps moyens et maximaux d'exécution (en secondes).

| prob | av (max) $\Delta UB$ | av $\Delta$ LB | # best | av (max) CPU |
|---|---|---|---|---|
| PAT | 0 (0) | - | 110/110 | 1.60 (59) |
| ALV | -0.46 (5.19) | 46.24 | 45/48 | 232.23 (498) |
| BL | 0.11 (4.17) | - | 38/39 | 17.61 (66) |
| KSD30 | 0 (0) | - | 480/480 | 10.26 (123) |
| KSD60 | 0.22 (3.54) | 10.81 | 413/480 | 38.78 (223) |
| KSD90 | 0.40 (5.65) | 10.29 | 379/480 | 61.25 (309) |
| KSD120 | 1.51 (20.91) | 32.41 | 241/600 | 207.93 (501) |

TAB. 2.1 – Résultats de l'approche proposée

Pour toutes les instances, excepté pour le jeu KSD120, les solutions retournées par LSSPER se situent en moyenne 0.5% au-dessus des meilleures solutions connues et nous avons amélioré la valeur de la meilleure solution connue pour certaines instances appartenant aux jeux ALV ou KSD. Nous possédons à ce jour 14, 9 et 4 meilleures solutions sur les jeux d'instances KSD60, KSD90 et KSD120, respectivement[2]. Toutes les instances de BL ont été résolues à l'optimum, exceptée une. Cependant les temps moyen d'exécution requis sont doubles de ceux dispensés à la résolution des instances de KSD30. Ceci semble confirmer la difficulté de ces instances hautement cumulatives. De façon générale, les temps d'exécution sont relativement élevés mais nous tenons cependant à

[2]les résultats sont disponibles sur http ://www.bwl.uni-kiel.de/Prod/psplib/library.html

souligner la simplicité, la facilité d'implémentation et la généralité de l'approche envisagée.

Pour démontrer de façon plus pertinente la généralité de notre schéma de résolution, nous avons tenté une approche de programmation linéaire pour la résolution des sous-problèmes. Ainsi, nous avons remplacé le solveur de contraintes par le solveur de programmation linéaire en nombres entiers ILOG Cplex. La recherche dans l'arbre se fait en profondeur d'abord et l'accent est mis sur la faisabilité. Le temps maximal alloué au solveur pour la résolution d'un sous-problème est fixé à 5 secondes. Les résultats obtenus sur les instances des jeux KSD30 et KSD60 sont reportés en Table 2.2. Ceux-ci montrent qu'en dépit de l'explosion des temps d'exécution, due au paramétrage, la déviation moyenne aux meilleures solutions connues est sensiblement la même que celle obtenue consécutivement à l'utilisation du solveur de contraintes. Ceci met en avant l'intérêt pratique de notre approche et tend à indiquer que le choix des sous-problèmes à résoudre est plus crucial que la méthode de résolution employée pour les résoudre, en regard de la qualité des solutions obtenues, tandis que la méthode de résolution influence quant à elle fortement le temps de calcul.

| prob | av (max) $\Delta$ UB | av $\Delta$ LB | # best | av CPU |
|---|---|---|---|---|
| KSD30 | 0.08 (3.17) | - | 457/480 | 165.04 (1485) |
| KSD60 | 0.63 (7.07) | 11.47 | 371/480 | 397.39 (2238) |

TAB. 2.2 – Résultats avec utilisation d'un solveur de PLNE pour la résolution des sous-problèmes

Cette assertion peut être vérifiée clairement par la Table 2.3 qui reporte les résultats obtenus sur le jeu d'instances KSD30 lors de l'utilisation de chacune des stratégies de génération des sous-problèmes. Les résultats sont éloquents : la stratégie par bloc dépasse grandement toutes les autres.

| méthode | $\Delta$ opt |
|---|---|
| Aléatoire&Critique | 1.5338 |
| Aléatoire&Prédécesseurs_Projet | 1.2370 |
| Aléatoire&Prédécesseurs_Ordo | 1.0885 |
| Aléatoire&Tous_Prédécesseurs | 0.9524 |
| Bloc | 0.02 |

TAB. 2.3 – Comparaison des stratégies de sélection sur le jeu KSD30

Finalement, nous avons testé l'efficacité séparée des différents composants de notre méthode. Ainsi, nous comparons les résultats obtenus par LSSPER aux deux heuristiques suivantes :

- Sched_alone : le solveur de contraintes (ILOG Scheduler) est utilisé comme méthode d'énumération implicite tronquée pour résoudre le problème global. Le schéma de branchement retenu est identique à celui utilisé lors de la résolution d'un sous-problème.
- FB_alone : la phase de résolution de sous-problème est retirée. La méthode consiste en une heuristique multi-start au cours de laquelle une procédure de forward-backward est appliquée itérativement.

Afin d'obtenir une comparaison équitable, ces deux méthodes sont stoppées lorsque le temps maximal d'exécution dispensé par LSSPER sur le même jeu d'instance est atteint. La Table 2.4 donne les résultats obtenus par Sched_alone, FB_alone et LSSPER sur les jeux d'instances KSD30 et KSD60.

| méthode | av (max) Δ UB | | av Δ LB | av (max) CPU | |
|---|---|---|---|---|---|
| | KSD30 | KSD60 | (KSD60) | KSD30 | KSD60 |
| Sched_alone | 0.17 (11.95) | 1.68 (20.13) | 13.03 | 12.77 (236) | 65.73 (427) |
| FB_alone | 0.32 (8.62) | 1.10 (16.03) | 12.12 | 123 (123) | 223 (223) |
| LSSPER | 0 (0) | 0.22 (3.54) | 10.81 | 10.26 (123) | 38.78 (223) |

TAB. 2.4 – Comparaison entre LSSPER, Sched_alone et FB_alone

Les résultats indiquent très nettement la différence de performances, tant au niveau des temps d'exécution que de la qualité des solutions obtenues, entre LSSPER et chacun de ses composants utilisés seuls. Ainsi, ils tendent à prouver que l'efficacité de LSSPER résulte de la coopération entre les différentes composantes : résolution exacte, recherche locale et heuristique de forward-backward.

Nous avons finalement voulu mesurer l'impact des caractéristiques des instances sur les performances de notre méthode. Dans les jeux d'instances KSD, ainsi que rapporté dans [Kolisch 2001], chaque instance est caractérisée par la densité du réseau (paramètre NC), le facteur de ressources (RF) représentant la consommation moyenne de ressources des activités, et la force des ressources (RS) qui varie proportionnellement à la capacité moyenne des ressources. Globalement, les instances hautement disjonctives, c.-à-d. telles que le paramètre RS est petit, semblent être les plus difficiles à résoudre. L'impact de ce paramètre est représenté en Figure 4. Nous pouvons constater de façon très claire que l'apparente difficulté du problème croît de façon exponentielle à mesure que le paramètre RS diminue, tandis que la variation des paramètres RS et NC ne semblent pas avoir le même effet, comme on peut l'apercevoir en Figures 2.11 et 2.12. Cependant, l'on peut constater la relative contre-performance de la méthode sur les instances pour lesquelles le paramètre RF vaut 0.25. Ceci semblerait indiquer que le processus de génération du sous-problème est moins adapté à ce cas particulier où une activité nécessite une ressource unique.

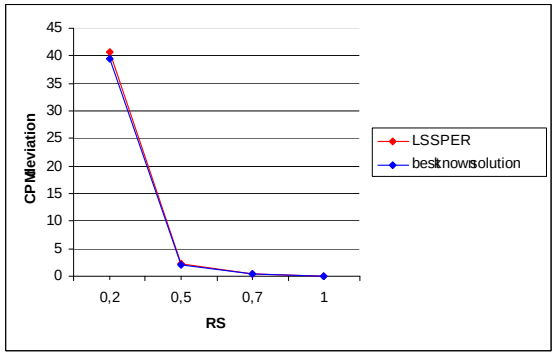


FIG. 2.10 – *Déviation selon le paramètre RS (KSD60)*

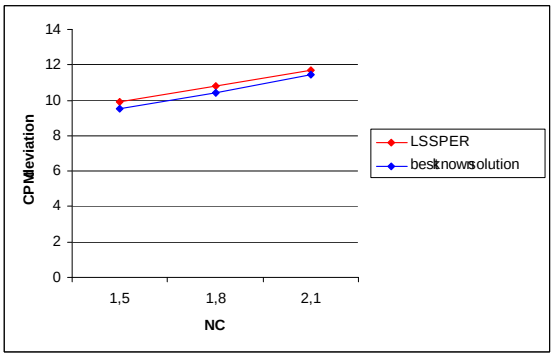


FIG. 2.11 – *Déviation selon le paramètre NC (KSD60)*

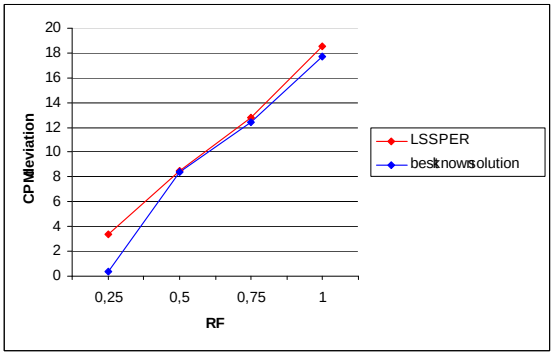


FIG. 2.12 – *Déviation selon le paramètre RF (KSD60)*

## 2.4 Une amélioration : la résolution en cascade

### 2.4.1 Principe

L'idée de l'amélioration proposée dans cette section repose sur la génération et la résolution de sous-problèmes consécutifs au cours d'une même itération. A une itération $s$, le problème global est alors décomposé en un sous-problème principal $P1$ et un sous-problème secondaire $P2$, qui est le sous-problème symétrique de $\Pi^s$. Celui-ci consiste à réordonnancer toutes les activités de $A \backslash A^s$, les activités de $A^s$ étant à leur tour fixées à leur valeur courante. L'idée consiste finalement à effectuer la post-optimisation de la résolution de $P1 = \Pi^s$ à l'aide de la méthode LSSPER elle-même. Ainsi, de même que précédemment, le sous-problème $P1$ est généré puis résolu. Si une solution est trouvée pour $P1$, alors le sous-problème résiduel $P2$ est à son tour généré puis résolu, en tenant compte du résultat de l'optimisation obtenu à l'issue de la résolution de $P1$. En raison de la taille importante de ce sous-problème résiduel (dans la plupart des cas), $P2$ est lui-même décomposé en $k$ sous-problèmes qui sont tour à tour résolus.

Les sections suivantes présentent les détails de la génération et de la résolution des sous-problèmes : le sous-problème principal $P1$ en section 2.4.2, le sous-problème résiduel $P2$ en section 2.4.3. Nous terminons ce chapitre par la présentation des résultats obtenus sur les jeux d'instances classiques en section 2.4.4.

### 2.4.2 Génération du sous-problème principal

La génération de $P1$ s'effectue de façon analogue à l'approche précédente : libération d'un ensemble $A_0^s$ de $p$ activités, remplacements des activités non sélectionnées par des fenêtres de temps sur les activités libérées et des périodes d'indisponibilité partielle des ressources. L'objectif est maintenant de déterminer les dates de début des activités de telle façon que la durée $\tilde{C}_{max} - \tilde{S}_{min}$ du sous-projet soit minimale. En effet, sachant que le sous-problème $P2$, consistant à réordonnancer les tâches de $A \backslash A^s$, sera résolu par la suite, il s'agit ici de "tasser" les tâches de $P1$ pour laisser le maximum de place à celles de $P2$. Le problème peut se formuler de la façon suivante :

$$
\begin{aligned}
(P1) \quad & \min \tilde{C}_{max} - \tilde{S}_{min} && && (2.11)\\
& \tilde{C}_{max} \geq S_i + p_i && \forall i \in A_0^s && (2.12)\\
& \tilde{S}_{min} \leq S_i && \forall i \in A_0^s && (2.13)\\
& S_j \geq S_i + p_i && \forall i, j \in A_0^s, (i,j) \in E && (2.14)\\
& ES_i \leq S_i \leq LF_i - p_i && \forall i \in A_0^s && (2.15)\\
& \sum_{i \in \mathcal{A}^s(t)} r_{ik} \leq R_k(t) && \forall k \in \mathcal{R}, \forall t && (2.16)
\end{aligned}
$$

### 2.4.3 Décomposition du sous-problème résiduel

Si une solution à $P1$ a été obtenue, alors le sous-problème $P2$ est généré en constituant les activités restantes en $k$ ensembles de taille $p$. Les ensembles sont divisés en deux types : les ensembles de successeurs et les ensembles de prédécesseurs. Ils sont construits de la façon suivante : l'ensemble de successeurs (resp. prédécesseurs) $A_l^s, l > 0$ est initialisé avec les successeurs (resp. prédécesseurs) dans le projet des activités déjà sélectionnées (appartenant au sous-problème principal $P1$ ou à l'un des ensembles $A_1^s, \ldots, A_{l-1}^s$ précédents). $A_l^s$ est ensuite complété à l'aide de la méthode de sélection par bloc. Les ensembles de successeurs sont déterminés en premier, puis les ensembles de prédécesseurs.

Une fois les ensembles construits, les sous-problèmes associés sont générés en remplaçant les activités n'appartenant pas à l'ensemble considéré par les contraintes appropriées. À ce moment-là, deux problèmes distincts complémentaires, décrits ci-dessous, sont à même d'être abordés. Pour les sous-problèmes associés aux ensembles de successeurs, il s'agit de déterminer un $p$-tuple de dates de début au plus tôt des activités de façon que la date de fin du sous-projet $\tilde{C}_{max}$ soit minimale ; un sous-problème associé à un ensemble de prédécesseurs consistera quant à lui à affecter une date de fin au plus tard à chacune des activités de telle sorte que la date de début du sous-projet soit maximale. Ces deux objectifs antagonistes tendent à resserrer la solution courante autour du

résultat de l'optimisation du sous-problème principal.

$$
\begin{aligned}
(SP\_succ) \qquad & \min \tilde{C}_{max} && && (2.17)\\
& \tilde{C}_{max} \geq S_i + p_i && \forall i \in A_l^s && (2.18)\\
& S_j \geq S_i + p_i && \forall i,j \in A_l^s, (i,j) \in E_l^s && (2.19)\\
& ES_i \leq S_i \leq LS_i && \forall i \in A_l^s && (2.20)\\
& \sum_{i \in \mathcal{A}_l^s(t)} r_{ik} \leq R_k(t) && \forall k \in \mathcal{R}, \forall t && (2.21)
\end{aligned}
$$

$$
\begin{aligned}
(SP\_pred) \qquad & \max \tilde{S}_{min} && && (2.22)\\
& \tilde{S}_{min} \leq C_i - p_i && \forall i \in A_l^s && (2.23)\\
& C_i \leq C_j - p_j && \forall i,j \in A_l^s, (i,j) \in E_l^s && (2.24)\\
& ES_i + p_i \leq C_i \leq LS_i + p_i && \forall i \in A_l^s && (2.25)\\
& \sum_{i \in \mathcal{A}_l^s(t)} r_{ik} \leq R_k(t) && \forall k \in \mathcal{R}, \forall t && (2.26)
\end{aligned}
$$

### 2.4.4 Résultats expérimentaux

Nous avons éprouvé l'approche proposée sur les mêmes jeux d'instances que précédemment. Le temps maximal d'exécution alloué au solveur pour la résolution d'un sous-problème a été fixé à 0,1 seconde.

Les résultats sont reportés dans le tableau 2.5. Pour chaque jeu d'instances, nous avons renseigné en colonnes 1 et 2 les pourcentages moyens et maximaux de déviation par rapport à la solution optimale (ou à la meilleure solution connue) et à la borne inférieure du chemin critique. Nous reportons en colonne 3 le nombre de concordances entre la meilleure solution répertoriée à ce jour et le résultat obtenu par la nouvelle version du programme. Nous indiquons finalement en dernière colonne les temps moyens et maximaux d'exécution.

| prob | av (max) $\Delta UB$ | av $\Delta$ LB | # best | av (max) CPU |
|---|---|---|---|---|
| PAT | 0 (0) | - | 110/110 | 0.69 (22) |
| ALV | -0.70 (0.94) | 45.83 | 45/48 | 165.42 (473) |
| BL | 0 (0) | - | 39/39 | 5.59 (19) |
| KSD30 | 0.02 (2.45) | - | 476/480 | 4.05 (67) |
| KSD60 | 0.21 (4.67) | 10.79 | 422/480 | 13.39 (82) |
| KSD90 | 0.44 (5) | 10.35 | 385/480 | 20.45 (116) |
| KSD120 | 1.31 (7.74) | 32.15 | 265/600 | 71.89 (158) |

TAB. 2.5 – Résultats de la version en cascade

Ces résultats montrent une amélioration globale des performances très légère, un peu plus marquée sur les instances à 120 activités, accompagnée d'une diminution considérable des temps d'exécution : ceux-ci se trouvent en moyenne divisés par un facteur de 2 !

Pour mesurer l'impact de la résolution en cascade par rapport à la résolution de sous-problèmes totalement indépendants les uns des autres, nous avons finalement testé la version originelle de notre méthode (un seul sous-problème résolu à chaque itération) sur les jeux d'instances KSD en limitant le temps de résolution d'un sous-problème à 0,1 seconde également. Pour une comparaison équitable, le processus est stoppé lorsque le nombre d'ordonnancements générés coïncide avec celui obtenu au terme de l'exécution de la version en cascade. Les diverses informations obtenues au terme de ces expérimentations sont présentées en Table 2.6.

| prob | av (max) $\Delta UB$ | av $\Delta$ LB | # best | av (max) CPU |
|---|---|---|---|---|
| KSD30 | 0.01 (2.06) | - | 476/480 | 6.20 (89) |
| KSD60 | 0.17 (3.06) | 10.74 | 422/480 | 33.21 (279) |
| KSD90 | 0.38 (5.88) | 10.25 | 389/480 | 63.99 (461) |
| KSD120 | 1.28 (4.81) | 32.07 | 245/600 | 270.90 (857) |

TAB. 2.6 – Résultats sur les jeux d'instances KSD

Une amélioration sensible en terme de qualité des solutions peut être notée. Elle est cependant largement amoindrie par l'explosion des temps de calcul. Dans le cadre des instances KSD120, celui-ci est en moyenne multiplié par un facteur de plus de 3,5 pouvant aller dans les cas les plus défavorable jusqu'à un facteur de presque 5,5 ! À nombre d'ordonnancements générés équivalent, la résolution exacte des sous-problèmes semble être donc mieux appréhendée lorsque les sous-problèmes successifs qui sont résolus possèdent un lien entre eux.

Afin d'établir une analyse plus pertinente de nos résultats, nous nous proposons de les comparer sur les jeux d'instance KSD avec les meilleures heuristiques actuelles : un algorithme à base de population proposé par Valls et al. [Valls 2004b], la méthode hybride de Kochetov et al. [Kochetov 2003b], l'algorithme génétique auto-adaptatif de Hartmann [Hartmann 2002], la méthode de recherche taboue de Nonobe et Ibaraki [Nonobe 2002] et le recuit simulé de Bouleimen et Lecocq [Bouleimen 2003]. Les résultats sont affichés dans la table 2.7 qui recense les déviations moyennes à l'optimum (KSD30) ou à la borne inférieure du chemin critique (KSD60, KSD120) pour chacune des approches.

| méthode | av $\Delta$ UB (KSD30) | av $\Delta$ LB (KSD60) | av $\Delta$ LB (KSD120) |
|---|---|---|---|
| LSSPER | 0.02 | 10.79 | 32.15 |
| Valls *et al.* [Valls 2004b] | 0.10 | 10.89 | 31.58 |
| Kochetov *et al.* [Kochetov 2003b] | 0.00 | 10.74 | 32.06 |
| Hartmann [Hartmann 2002] | 0.22 | 11.70 | 35.39 |
| Nonobe, Ibaraki [Nonobe 2002] | - | - | 35.86 |
| Bouleimen, Lecocq [Bouleimen 2003] | 0.23 | 11.90 | 37.68 |

TAB. 2.7 – Comparaison avec les meilleures heuristiques actuelles

Ces résultats semblent exhiber l'efficacité de la méthode que nous proposons en terme de qualité des solutions obtenues. En effet, ils outrepassent la plupart de ceux obtenus par les autres heuristiques, à l'exception des méthodes proposées par Valls *et al.* et Kochetov *et al.* qui obtiennent des

résultats équivalents voire meilleurs sur certains jeux d'instances en des temps d'exécution bien moindres.

Il serait néanmoins intéressant de voir quelles améliorations pourraient encore être amenées en jouant sur les deux points cruciaux de notre méthode, à savoir :

- le procédé de génération des sous-problèmes pour espérer améliorer la qualité des résultats ;
- la méthode de résolution des sous-problèmes pour réduire les temps d'exécution.

En ce qui concerne le premier point, nous savons à présent que l'indépendance des sous-problèmes par rapport au problème global, ainsi que le lien qu'ils peuvent avoir entre eux, jouent un rôle déterminant pour l'obtention de solutions de qualité. Il s'agirait ainsi de penser de nouvelles façon de générer les sous-problèmes qui respectent ces deux caractéristiques. Une piste avancée par l'analyse des résultats serait de développer une stratégie de libération de blocs d'activités s'exécutant sur la même ressource, ce qui pourrait être plus efficace sur les instances telles que RF=0,25. Cependant, ce choix pourrait se révéler non avantageux pour les instances présentant une valeur de ce paramètre différente. Il pourrait alors être intéressant de se pencher sur la conception d'une méthode auto-adaptative qui sélectionnerait elle-même le schéma de génération de sous-problèmes le plus approprié aux paramètres de l'instance à résoudre.

Une deuxième visée concerne la diminution des temps d'exécution de la méthode, car ils constituent à l'heure actuelle le point critique de l'approche. Il s'agirait ici de concevoir d'autres façons de parcourir le voisinage. Nous pourrions ainsi remplacer la recherche arborescente par une procédure plus rapide, telle un algorithme de recherche locale ou encore une recherche arborescente partielle comme LDS.

# Chapitre 3

# Application de LSSPER au problème d'affectation de fréquences

*Ce chapitre traite de l'application de LSSPER à un problème d'affectation de fréquences (noté FAP pour Frequency Assignment Problem) dans un réseau de télécommunications par voie hertzienne. L'intérêt est ici de démontrer la validité de notre approche sur des instances réalistes, issues d'applications militaires. De fait, à l'inverse des formulations classiques qui seront présentées en section 3.1, et qui reposent sur des hypothèses simplificatrices, le problème étudié ici fait intervenir un nouveau type de contrainte permettant la prise en charge de sources d'interférences multiples. Nous présenterons ces contraintes additionnelles en section 3.1.3. De par l'aspect novateur de cette nouvelle formulation, peu de recherches ont été réalisées sur le sujet et nous présenterons donc en section 3.2 les concepts de diverses méthodes heuristiques appliquées à des formulations plus classiques de ce problème. Nous poursuivrons en section 3.3 par l'application de notre méthode à ce problème particulier puis terminerons par des résultats expérimentaux en section 3.3.8, que nous comparerons à ceux obtenus par deux autres approches éprouvées sur ce problème particulier. Nous tenterons également au cours de cette section de souligner le bénéfice apporté par cette nouvelle formulation.*

## 3.1 Description du problème

### 3.1.1 Définition générale des problèmes d'affectation de fréquences

Un réseau de télécommunications par voie hertzienne est constitué d'un ensemble de *sites* où sont localisés les supports de transmission, à savoir les antennes reliées à des *émetteurs/récepteurs*. Des *liaisons*, composées d'un ou plusieurs *trajets*, sont établies entre des sites géographiques distincts comme représenté en Figure 3.1.

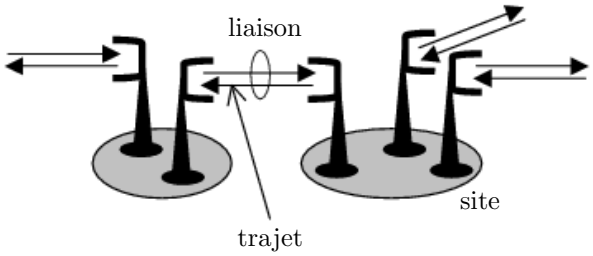


FIG. 3.1 – *Sites, liaisons et trajets*

Chaque trajet $i \in T = \{1, \ldots, n\}$, qui consiste en un bond radioélectrique unidirectionnel établi entre deux antennes, est associé à un *système* $\sigma(i)$ déterminant le type de matériel utilisé ainsi que ses caractéristiques d'émission et de réception. Il faut lui affecter une *ressource* $(f_i, p_i)$ dont les composantes correspondent respectivement à la *fréquence* porteuse du signal transmis et à la *polarisation* de l'onde. Dans le cadre de la présente étude cependant, le cas d'une polarisation unique sera considéré, ce qui revient à considérer exclusivement la composante fréquentielle de la ressource. Par conséquent, le problème revient à caractériser le $n$-uplet $F = \{f_1, \ldots, f_n\}$ représentant l'affectation à chaque trajet $i$ d'une fréquence de son *domaine* $D_i$. Les domaines des trajets sont généralement déterminés par la réglementation, les contraintes inhérentes au matériel, les choix du gestionnaire de fréquences, etc. . . .

Cependant, les fréquences ne peuvent être allouées aux trajets sans précaution. En effet, la localisation, sur un même site, de plusieurs émetteurs/récepteurs impose des limitations quant aux ressources affectées afin de garantir un fonctionnement optimal, ce qui se traduit par le respect d'un certain nombre de contraintes, de natures diverses. On distingue par exemple :

- les contraintes simples d'*égalité* entre fréquences ;
- les contraintes simples de *différence* entre fréquences ;
- les contraintes "*duplex*" ;
- les contraintes de *pré-affectation* ;
- les contraintes de *compatibilité électromagnétique*.

On dit d'une solution qu'elle est *réalisable* lorsqu'elle satisfait à l'ensemble des contraintes énoncées ci-dessus. Le problème consiste alors à optimiser un objectif donné (nous en présenterons quelques uns au cours des sections suivantes). Malheureusement, nombre de problèmes réels, de par

des exigences trop fortes ou des domaines fréquentiels trop restreints, ne possèdent pas de solution admissible. Il s'avère de fait nécessaire d'offrir la possibilité de rechercher une solution *dégradée* proposant une perte de qualité acceptable. Pour cela, l'approche classique consiste à partitionner les contraintes en deux sous-ensembles :

– les contraintes *dures*, nécessairement satisfaites, englobant les contraintes de type simples et duplex, regroupées sous la dénomination de contraintes *impératives* d'égalité ou de différence. Les premières imposent un écart $\epsilon_{ij}$, nul dans le cas des contraintes simples, sur les fréquences allouées à deux trajets $i$ et $j$ donnés (constitutifs d'une même liaison dans le cas des contraintes duplex) :

$$|f_i - f_j| = \epsilon_{ij} \quad \epsilon_{ij} \geq 0 \tag{3.1}$$

Les contraintes de différence interdisent pour leur part à deux trajets $i$ et $j$ d'être affectés à la même fréquence :

$$|f_i - f_j| \neq 0 \tag{3.2}$$

– les contraintes *souples*, de types pré-affectation ou compatibilité électromagnétique, dont la violation, autorisée, a pour effet une diminution de la qualité de transmission. Les premières traduisent la nécessité d'allouer une fréquence pré-déterminée $fp_i$ à un trajet $i$ :

$$f_i = fp_i \tag{3.3}$$

Les deuxièmes imposent un écart minimum $\delta_{ij}$ sur les fréquences allouées à deux trajets $i$ et $j$ donnés :

$$|f_i - f_j| \geq \delta_{ij} \tag{3.4}$$

L'objectif du problème consiste alors à rechercher une affectation des fréquences induisant le *minimum d'interférences* (MI-FAP pour Minimum Interference) dans le réseau de télécommunications, ce qui se traduit par la minimisation de la *somme pondérée* des contraintes souples violées :

$$min \sum q_{ij}(|f_i - f_j| \leq \delta_{ij}) + \sum m_i(|f_i - fp_i| > 0) \tag{3.5}$$

où $q_{ij}$ et $m_i$ représentent respectivement le coût de violation de la contrainte d'écart minimum et de la contrainte de pré-affectation associées et $(c)$ vaut 1 si la condition $c$ est vérifiée, 0 sinon.

### 3.1.2 Graphe de contraintes et complexité

Le FAP peut être représenté par un *graphe de contraintes* $G = (T, E)$, également appelé *graphe d'interférences*. Les sommets sont les trajets du réseau, les arcs relient deux trajets impliqués dans une même contrainte. La figure 3.2 représente le graphe d'interférences induit par les contraintes impératives et de compatibilité électromagnétique d'une des instances qui sont l'objet de la présente étude.

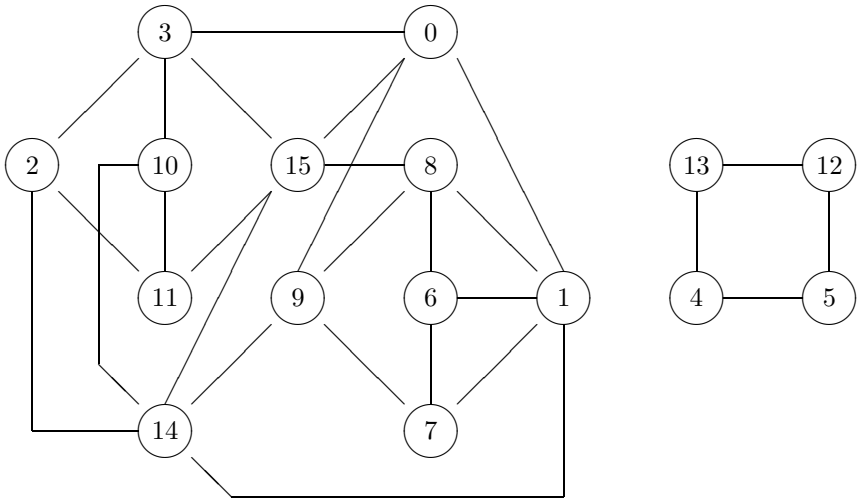


FIG. 3.2 – *Graphe de contraintes partiel de l'instance FAPPG01_0016*

En particulier, supposant qu'il n'y ait que des contraintes d'écart minimum, que les écarts soient unitaires, et que les domaines fréquentiels des trajets soient tous identiques et égaux à l'ensemble des entiers entre 1 et $k$, la question de savoir s'il existe une solution qui satisfasse toutes les contraintes revient à rechercher une *$k$-coloration valide* du graphe $G$, ce qui est un problème NP-complet pour $k \geq 3$.

### 3.1.3 Une formulation plus réaliste des contraintes électromagnétiques

**Contexte**

La problématique étudiée dans le présent chapitre a été proposée par le Centre ELectronique de l'ARmement (CELAR) qui est une antenne de la Délégation Générale pour l'Armement (DGA) spécialisée notamment dans le suivi et la réalisation de travaux visant à optimiser l'utilisation du spectre de fréquences au sein des forces armées françaises. Dans ce contexte, le projet proposé se rattache à celui d'affectation de fréquences dans des réseaux de télécommunications par voie hertzienne. Dans le même cadre, divers sujets ont auparavant été proposés. On peut notamment se référer au projet CALMA (Combinatorial ALgorithms for Military Applications) qui s'est constitué dans les années 1993-1995, ou encore au challenge ROADEF 2001 qui portait sur des instances de FAP avec polarisation. Les problèmes présentés dans la suite de ce chapitre se focalisent pour leur part sur l'enrichissement du modèle introduit au cours de la section précédente afin de prendre en compte la notion de contraintes de compatibilité électromagnétique *cumulatives*. Cette étude correspond à un appel d'offre où plusieurs équipes sont intervenues et ont travaillé sur des approches de résolution diverses. Ainsi, un algorithme tabou basé sur le principe de voisinage consistant a été développé par Vlasak et Vasquez [Vlasak 2003], une procédure de recuit simulé par Sarzeaud [Sarzeaud 2003] et une méthode exacte hybride PPC-PLNE par Oliva [Oliva 2003].

Notre propre contribution porte sur l'application de notre méthode de grands voisinages. Il s'agira ainsi que précédemment de spécifier et adapter au mieux les points cruciaux, notamment la génération et résolution des divers sous-problèmes.

**Contraintes de compatibilité électromagnétique : insuffisances et contraintes globales**

L'affectation des fréquences aux trajets doit se baser sur l'environnement électromagnétique de la zone et prendre en compte toutes les perturbations susceptibles d'induire des interférences dans les communications. Cet environnement est composé de systèmes susceptibles d'être de nature différente et plusieurs équipements peuvent être amenés à communiquer à proximité les uns des autres.

Dans ce contexte hétérogène, la qualité de communication d'un récepteur donné est déterminée par des calculs d'ingénierie de liaison et de compatibilité électromagnétique qui prennent en compte les émissions générées par les émetteurs voisins susceptibles de le perturber. Une mesure courante de cette qualité est donnée par un rapport $C/I$ qui fournit un seuil limite à ne pas dépasser entre la puissance utile $C$ du trajet potentiellement perturbé et la puissance perturbatrice $I$ émanant de tous les émetteurs avoisinants.

Une hypothèse simplificatrice est souvent retenue par les approches classiques qui considèrent que ce "droit à perturber" est réparti de façon *uniforme* entre les différents perturbateurs potentiels. On passe donc d'une situation complexe avec plusieurs perturbateurs sur un récepteur à une situation largement simplifiée consistant en plusieurs situations élémentaires à un perturbateur et un récepteur (voir Figure 3.3). Cette simplification permet d'obtenir les contraintes *binaires* de compatibilité électromagnétique présentées en section 3.1.1.

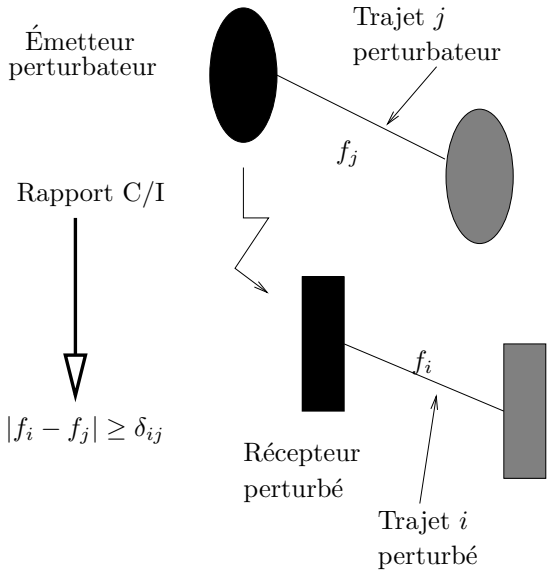


FIG. 3.3 – *Une situation élémentaire avec un perturbateur et un récepteur*

Cette formulation ne possède néanmoins pas de justification réelle et a de plus pour effet de

surcontraindre les affectations des fréquences aux trajets. C'est pourquoi nous allons à présent introduire un type de contraintes additionnel qui va permettre de lever cette hypothèse d'uniformité. Les contraintes binaires classiques vont ainsi être remplacées par des contraintes plus souples, mais aussi plus complexes, qui considèrent de façon *simultanée* les fréquences allouées à tous les trajets perturbateurs et au trajet perturbé. La Figure 3.4 illustre le cas de perturbateurs multiples.

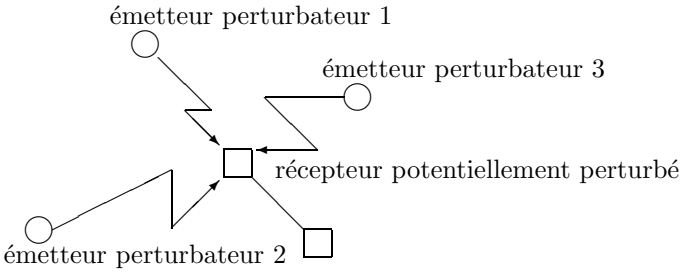


FIG. 3.4 – *Perturbations multiples sur un récepteur*

Soit $P_i$, l'ensemble des perturbateurs d'un trajet $i$, $T_{\sigma(i)\sigma(j)|f_i-f_j|}$, la perturbation induite par le trajet $j$ (de système $\sigma(j)$) sur le trajet $i$ (de système $\sigma(i)$) lorsque les fréquences affectées à $i$ et $j$ sont $f_i$ et $f_j$, respectivement, et $\Lambda_i$, le seuil acceptable d'interférences calculé à partir du rapport $C/I$. On exprime alors la perturbation globale exercée sur le trajet $i$ de la façon suivante :

$$\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)|f_i-f_j|} \leq \Lambda_i \tag{3.6}$$

L'influence exercée par un trajet perturbateur $j$ est ici pondérée par un facteur multiplicatif $\lambda_{ij}$ qui prend en compte, entre autres, la distance et les orientations respectives des trajets considérés. La fonction $T_{\sigma(i)\sigma(j)}$ est positive, décroissante et tend vers 0 à mesure que l'écart fréquentiel $|f_i - f_j|$ augmente. La figure 3.5 présente un exemple de fonction $T_{\sigma(i)\sigma(j)}$.

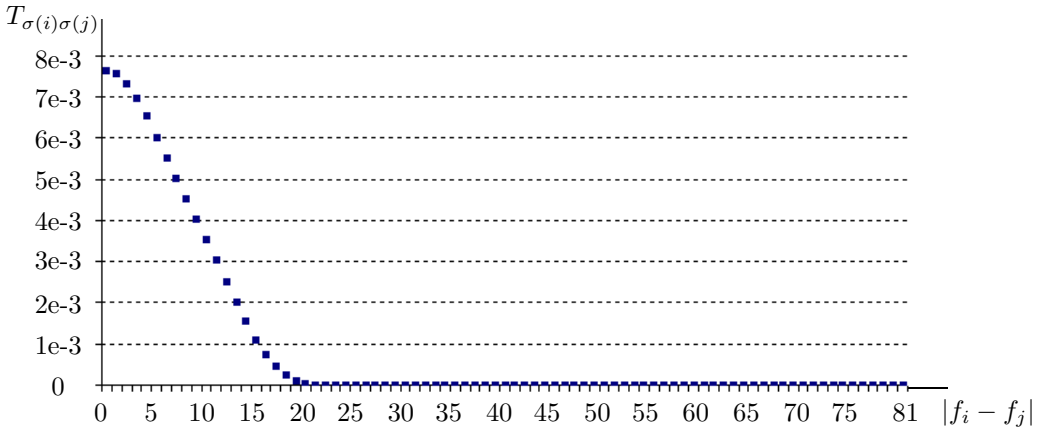


FIG. 3.5 – *Fonction tabulée* $T_{\sigma(i)\sigma(j)}$

**Problèmes étudiés**

Nous nous proposons de considérer ce nouveau type de modélisation, plus réaliste, abordé récemment dans la littérature par quelques études de Dunkin *et al.* [Dunkin 1998] ou Mannino et Sassano [Mannino 2003]. De façon plus précise, deux types de contraintes de compatibilité électromagnétique sont prises en compte :

- dans le cas d'émetteurs perturbateurs co-localisés avec le récepteur perturbé, la forme classique binaire est conservée car ces perturbations, dites de "*champ proche*", sont prises en compte sous la forme de contraintes forfaitaires (contraintes *co-site*).
- dans le cas d'émetteurs perturbateurs plus éloignés (perturbations de type "*champ lointain*"), la nouvelle formulation est considérée (contraintes *globales*).

L'objectif consiste alors dans un premier temps à rechercher la réalisabilité du problème, c.-à-d. le respect du maximum de contraintes souples, idéalement toutes. L'optimisation porte ainsi sur la minimisation de la somme pondérée des contraintes co-site et globales violées, les instances ne comportant pas de contraintes de pré-affectation. Dans le modèle que nous considérons, les pondérations ne diffèrent qu'avec le type de contraintes (i.e. deux contraintes de même type sont pondérées de la même façon). Soit $G_{IE} = (T, E_{IE})$ (resp. $G_{ID} = (T, E_{ID})$, $G_C = (T, E_C)$, $G_G = (T, C_G)$), le sous-graphe de $G$ induit par les contraintes impératives d'égalité (resp. impératives de différence, co-site, globales), et $C_i \in C_G$ la clique associée à la contrainte globale de trajet perturbé $i$, ce problème peut s'exprimer de la façon suivante :

$$(MI-FAP) \qquad g = \min \alpha|V| + \beta|W| \tag{3.7}$$

$$f_i \in D_i \qquad \forall\, i \in T \tag{3.8}$$

$$|f_i - f_j| = \epsilon_{ij} \qquad \forall\, (i,j) \in E_{IE} \tag{3.9}$$

$$|f_i - f_j| \neq \epsilon_{ij} \qquad \forall\, (i,j) \in E_{ID} \tag{3.10}$$

$$V = \{|f_i - f_j| < \delta_{ij}, (i,j) \in E_C\} \tag{3.11}$$

$$W = \{\textstyle\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)|f_i - f_j|} > \Lambda_i, C_i \in C_G\} \tag{3.12}$$

Si une solution satisfaisant l'ensemble des contraintes de compatibilité électromagnétique est découverte, une deuxième phase est initiée. Il s'agit alors de minimiser la *largeur de bande passante* occupée (MS-FAP pour Minimum Span), c.-à-d. la différence entre la plus grande et la plus petite fréquence affectée. Ce deuxième problème peut être formulé comme suit :

$$(MS-FAP) \qquad h = \min(\max f_i - \min f_i) \tag{3.13}$$

$$f_i \in D_i \qquad \forall\, i \in T \tag{3.14}$$

$$|f_i - f_j| = \epsilon_{ij} \qquad \forall\, (i,j) \in E_{IE} \tag{3.15}$$

$$|f_i - f_j| \neq \epsilon_{ij} \qquad \forall\, (i,j) \in E_{ID} \tag{3.16}$$

$$|f_i - f_j| \geq \delta_{ij} \qquad \forall\, (i,j) \in E_C \tag{3.17}$$

$$\textstyle\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)|f_i - f_j|} \leq \Lambda_i \quad \forall\, C_i \in C_G \tag{3.18}$$

**Les instances du CELAR**

A partir de données récoltées sur le terrain, le CELAR a généré 30 instances pour le problème d'affectation de fréquences avec sommation des perturbateurs, intitulées FAPPG$x_n$. Celles-ci comportent :

- de 16 à 2454 trajets ;
- de 10 à 1229 contraintes impératives de type (3.1) et (3.2) ;
- de 16 à 4155 contraintes binaires co-site de type (3.4) ;
- de 16 à 2015 contraintes cumulatives globales de type (3.6).

Les caractéristiques de chaque instance sont reportés dans le Tableau 3.1.

| FAPPG(1) | #CI(2) | #CE(3) | #CC(4) | FAPPG(1) | #CI(2) | #CE(3) | #CC(4) |
|---|---|---|---|---|---|---|---|
| 01_0016 | 10 | 16 | 16 | 16_0038 | 21 | 128 | 37 |
| 02_0018 | 11 | 24 | 18 | 17_0040 | 22 | 92 | 36 |
| 03_0066 | 35 | 100 | 50 | 18_0052 | 28 | 116 | 42 |
| 04_0064 | 34 | 88 | 48 | 19_0770 | 387 | 2276 | 770 |
| 05_0064 | 34 | 80 | 64 | 20_1930 | 967 | 3896 | 1075 |
| 06_0182 | 93 | 245 | 172 | 21_1088 | 546 | 2789 | 1081 |
| 07_0182 | 93 | 245 | 172 | 22_0768 | 386 | 1604 | 757 |
| 08_0608 | 306 | 812 | 484 | 23_0034 | 19 | 53 | 24 |
| 09_1460 | 732 | 1862 | 1123 | 24_0048 | 38 | 80 | 39 |
| 10_1698 | 851 | 705 | 1292 | 25_0106 | 68 | 181 | 63 |
| 11_0164 | 84 | 439 | 101 | 26_0140 | 85 | 219 | 83 |
| 12_0902 | 453 | 2354 | 572 | 27_0154 | 92 | 255 | 134 |
| 13_0306 | 155 | 1142 | 244 | 28_0398 | 199 | 821 | 340 |
| 14_0194 | 99 | 587 | 163 | 29_0526 | 263 | 980 | 471 |
| 15_2454 | 1229 | 5135 | 2015 | 30_2166 | 1083 | 4155 | 1985 |

(1) nom du problème FAPPG$x_n$, où $x$ dénote le numéro du problème et $n$ représente le nombre de trajets.
(2) nombre de contraintes impératives d'égalité et de différence
(3) nombre de contraintes co-site
(4) nombre de contraintes globales

TAB. 3.1 – Les instances générées par le CELAR

La taille importante de certaines de ces instances, ainsi que le nombre élevé de contraintes qu'elles font intervenir, justifient l'utilisation d'une metaheuristique pour résoudre ce problème. On peut en effet se référer à [Oliva 2003] pour la présentation d'une méthode exacte qui souligne les limitations de ce type d'approche sur ces problèmes.

Le nombre de variables pouvant atteindre 2500, il est de plus intéressant d'étudier comment notre méthode de grands voisinages se comporte face à cet important changement d'échelle (pas plus de 120 activités pour les problèmes de RCPSP) et quelles sont les adaptations nécessaires à son application à un cas réaliste.

## 3.2 Heuristiques et voisinages pour le FAP

Dans cette partie, nous allons présenter diverses heuristiques ayant été appliquées à des problèmes d'affectation de fréquences particuliers, notamment sur les instances CALMA. Différentes problématiques peuvent être rencontrées : outre le MI-FAP et MS-FAP présentés précédemment, certains problèmes envisagent également la minimisation de la probabilité moyenne de *blocages* des transmissions dans le réseau (MB-FAP pour Minimum Blocking) ou encore la minimisation du *nombre de fréquences* utilisées (MO-FAP pour Minimum Order). Nous focalisons cependant notre revue de la littérature sur les deux premiers types de problèmes, puisqu'ils sont, exclusivement l'objet de notre travail. Pour une vision plus complète, nous recommandons la lecture de [Aardal 2001].

Dans les parties suivantes, nous présentons les méthodes qui nous apparaissent comme étant les plus caractéristiques des différentes approches heuristiques existantes en suivant un plan similaire à celui du chapitre précédent, à savoir :

- en section 3.2.1, nous présentons quelques méthodes de type constructif ;
- la section 3.2.2 est dévolue à la présentation d'algorithmes génétiques et d'optimisation par colonies de fourmis ;
- les procédures de recherche locale et les divers voisinages qu'elles font intervenir sont présentés en section 3.2.3 ;
- nous terminons en section 3.2.4 par l'exposé d'une méthode hybride ad-hoc pour la résolution de ce problème particulier.

### 3.2.1 Les heuristiques constructives

Ce premier type de méthodes se base sur un schéma glouton. La décision qui est réalisée à une itération donnée est irrévocable. Elle résulte d'un double choix : sélection du prochain trajet à instancier et détermination de la fréquence à laquelle il sera finalement affecté. Pour une heuristique constructive donnée, ll s'agit alors de spécifier ces deux points.

Dans les travaux de Zoellner et Beall [Zoellner 1977], dévolus à la résolution de problèmes de type MS-FAP, différents algorithmes gloutons sont décrits puis comparés. Trois ordres possibles de sélection des trajets, basés sur des considérations relatives au graphe d'interférences, sont ainsi considérés de même que deux possibilités distinctes d'assignation des fréquences. Dans le cadre de problèmes de MI-FAP, un de ces ordres de sélection, consistant à instancier le sommet possédant le degré le plus important dans le graphe des contraintes (*highest degree first*), est repris par Sivarajan et al. [Sivarajan 1989] sous une variante, permettant la prise en compte des caractéristiques particulières des instances Philadelphia.

D'autres auteurs ont cherché à exploiter l'analogie entre FAP et problème de *coloration de graphes* (GCP pour Graph Coloring Problem) en généralisant la célèbre procédure DSATUR mise au point par Brélaz [Brélaz 1979]. Celle-ci fait intervenir la notion de *degré de saturation* d'un sommet. Pour des problèmes de FAP, ce degré est donné par le nombre de fréquences dont l'affectation au trajet considéré entraînerait la violation de contraintes dures. Dans la version originelle de l'algorithme, reprise telle quelle par Costa [Costa 1993] pour la résolution du MO-FAP, le trajet

possédant le degré de saturation le plus élevé est sélectionné puis affecté à la fréquence disponible la plus petite possible. De légères adaptations peuvent être rencontrées afin de traiter des problèmes de type MI-FAP : affectation de la fréquence minimisant l'augmentation de la valeur de fonction objectif (Borndörfer et al. [Borndörfer 1998]), degré de saturation généralisé (Generalised Saturation Degree), introduit par Valenzuela et al. [Valenzuela 1998], calculé par le biais des pénalités associées aux contraintes que l'affectation des fréquences interdites amènerait à violer. La procédure d'*empaquetage séquentiel généralisé* (Generalised Sequential Packing) mise au point par Sung et Wong [Sung 1997], s'appuie également sur la connection entre GCP et FAP pour résoudre des problèmes d'affectation de canaux : à une itération donnée, l'algorithme tente de colorier, selon un procédé bien défini, le maximum de sommets du graphe d'interférences avec les couleurs $c_i$ et $c_{i+1}$ (équivalent à assigner les canaux $c_i$ et $c_{i+1}$ aux différentes cellules du système). Le processus est répété pour les couleurs suivantes jusqu'à ce qu'une solution soit trouvée.

### 3.2.2 Les méthodes évolutionnistes

#### Les algorithmes génétiques

Diverses représentations des solutions peuvent être rencontrées au gré des travaux publiés sur ce problème particulier. Une première d'entre elles consiste en un encodage direct de la solution par un *vecteur d'affectation des fréquences* aux trajets. Une deuxième représentation se base sur le *partitionnement* des trajets en sous-groupes, chaque sous-groupe comprenant l'ensemble des trajets affectés à une fréquence donnée. Finalement, une dernière représentation envisage une *liste de trajets*. L'obtention d'une solution est ici réalisée en appliquant un processus d'*affectation canonique* des fréquences, c.-à-d. en affectant les trajets dans l'ordre indiqué par la liste à la plus petite fréquence disponible.

C'est cette dernière représentation qui est utilisée dans les travaux de Valenzuela [Valenzuela 1998] dévolus à la résolution de problèmes de type MS-FAP. Les permutations composant la population initiale sont générées de façon aléatoire puis modifiées en appliquant le critère de sélection de l'algorithme GSD. Cette population est par la suite soumise aux différents opérateurs d'évolution. La descendance est générée en testant divers opérateurs de croisement (de *position* ou d'*ordre*) dont le plus efficace semble être l'opérateur de cycle[1]. De même, divers opérateurs de mutation sont éprouvés parmi lesquels l'opérateur de position (sélection de deux trajets et déplacement du deuxième immédiatement après le premier dans la liste) obtient les meilleurs résultats. Le processus de sélection consiste à comparer la solution correspondant à chaque enfant nouvellement produit à celle associée au "pire" de ses parents et à remplacer ce dernier dans la génération suivante par son enfant si celui-ci s'avère de meilleure qualité.

Une approche plus originale, testée sur des instances de MI-FAP, a été introduite par Kolen [Kolen 1999]. Celle-ci est basée sur une définition résolument innovante des opérateurs de croisement et de mutation. En effet, ceux-ci réalisent des opérations d'optimisation sur les individus de la population. L'opérateur de croisement génère à partir de deux parents un enfant unique qui consiste en la meilleure combinaison de leurs gènes. Plus précisément, chaque trajet de l'enfant

[1] $A = \underline{6}4\underline{1}25\underline{3}, B = \overline{1}5\overline{3}24\overline{6} : A' = \underline{6}5\underline{1}24\underline{3}, B' = \overline{1}4\overline{3}25\overline{6}$

héritera sa fréquence d'affectation soit de son père, soit de sa mère. L'ensemble des combinaisons possibles est calculé en appliquant une procédure de branch-and-cut basée sur une formulation partiellement contrainte du MI-FAP. L'opérateur de mutation définit quant à lui un voisinage de type 1-opt et le parcourt de façon exhaustive afin de générer une nouvelle solution 1-optimale. Il est appliqué à chaque individu nouvellement produit de la population.

D'autres approches, basées sur les deux premiers types de représentation, peuvent être rencontrées : dans le cadre de leurs travaux, Crompton et al. [Crompton 1994] comparent les deux formulations, associées chacune aux opérateurs de croisement et de mutation appropriés, et concluent sur l'efficacité supérieure de la représentation par partitionnement sur la représentation directe.

#### Colonies de fourmis

Pour la résolution de MI-FAP, Manezzio et Carbonaro [Manezzio 2000] présentent la première application d'une procédure basée sur les colonies de fourmis, qu'ils nomment ANTS pour Approximate Nondeterministic Tree-Search Procedure. Celle-ci incorpore à un processus d'optimisation classique par colonies de fourmis des fonctionnalités supplémentaires comme l'utilisation de bornes inférieures ou encore des mécanismes de non stagnation des solutions. Chaque fourmi bâtit une solution en étendant itérativement une affectation partielle par le choix d'un nouveau trajet à affecter et de sa fréquence d'assignation. Le choix du mouvement réalisé par la fourmi est facteur de la quantité de phéromone associé à celui-ci mais aussi d'informations heuristiques a priori. Cette procédure est couplée à une technique d'optimisation locale qui est appliquée à chaque solution produite.

Montemanni et al. [Montemanni 2002] adoptent eux aussi ce paradigme pour résoudre des problèmes de MS-FAP prenant en considération des perturbations multiples. Leur stratégie consiste à obtenir pour une valeur suffisamment élevée de largeur de bande une affectation sans interférences à l'aide d'une procédure de type ANTS puis à décrémenter cette valeur de largeur de bande. Le processus est alors itéré jusqu'à l'obtention d'un critère d'arrêt. L'optimisation est ici encore couplée à l'utilisation d'une procédure d'amélioration de recherche locale.

### 3.2.3 Les approches de recherche locale

La plupart des méthodes rencontrées dans la littérature utilisent le voisinage 1-échange qui comporte toutes les solutions qui diffèrent de la solution courante par une fréquence d'assignation d'un trajet. Ce voisinage est invoqué tel quel dans la procédure taboue mise au point par Castelino et al. [Castelino 1996] ou encore, dans celle de Capone et Trubian [Capone 1999]. Ces derniers font intervenir le calcul d'une fonction objectif simplifiée préalablement au parcours du voisinage afin de supprimer d'emblée des mouvements non intéressants.

Dans de nombreux travaux, ce voisinage n'est cependant pas parcouru de manière exhaustive. Ainsi dans [Hao 1998] puis [Hao 1999], Hao *et al.* restreignent leur voisinage en réalisant une sélection sur les trajets à considérer pour le mouvement : sous-ensemble généré de façon aléatoire puis ensemble des sommets impliqués dans des violations de contraintes. Bouju et al. [Bouju 1995] prennent également le parti de ne considérer que les solutions atteignables en réaffectant l'un des $k$ trajets possédant les plus grandes violations. Ce paramètre $k$ est incrémenté au fur et à mesure

du déroulement du processus. D'autres approches vont également dans ce sens. Dans [Costa 1993], Costa restreint le voisinage de sa méthode taboue en fixant un nombre maximal de tentatives de mouvement. Les trajets possédant les plus grandes violations sont choisis en premier. Dans ce même papier, il présente également une approche de recuit simulé dans laquelle il ne considère la réaffectation que d'un trajet bien déterminé, choisi parmi les trajets qui interviennent dans une contrainte violée. Dans Castelino et al. [Castelino 1996], une méthode de recherche locale simple, utilisée comme procédure élémentaire, réalise des passes multiples constituées de $n$ itérations, chaque itération $i$ amenant à parcourir le voisinage 1-échange restreint au trajet $i$. D'autres auteurs restreignent encore davantage leur voisinage en considérant la réaffectation d'un seul trajet à une fréquence déterminée. Dans la procédure de recuit simulé de Knälmann et al. [Knälmann 1994], le trajet choisi ainsi que sa fréquence de réaffectation sont déterminés de façon aléatoire. Sarzeaud [Sarzeaud 2003] utilise quant à lui des techniques d'échantillonnage de Gibbs pour le choix du trajet et de sa nouvelle fréquence d'affectation. Beckmann et Killat [Beckmann 1999] proposent quant à eux de réassigner un trajet sélectionné de façon aléatoire à la fréquence qui minimise les interférences. Ce type de voisinage se retrouve également dans l'approche développée par Zerovnik [Zerovnik 1997] : un trajet, sélectionné parmi ceux impliqués dans de nombreuses contraintes violées, est affecté à une fréquence donnée avec une probabilité dépendant du niveau d'interférences que l'assignation effective induirait.

L'algorithme de Park et Lee [Park 1996] emploie quant à lui un voisinage de type 2-échange, c.-à-d. comprenant toutes les solutions pouvant être obtenues à partir de la solution courante en sélectionnant deux trajets et intervertissant leurs fréquences d'affectation. Un autre voisinage faisant intervenir la réaffectation de deux sommets est invoqué dans la procédure de type *recherche locale guidée* (Guided Local Search) mise au point par Tsang et Voudouris [Tsang 1998]. L'optique est de considérer toutes les affectations nouvelles possibles d'une paire de trajets "couplés" par une contrainte. Le voisinage de taille possiblement élevé défini par ce mouvement impose l'utilisation d'une procédure de *recherche restreinte* du voisinage (Fast Local Search).

D'autres types de voisinage sont induits par des modifications réalisées sur une liste de trajets. Une solution peut être obtenue à partir d'une telle liste par application d'une procédure d'affectation canonique. Un mouvement peut ainsi résulter de l'échange de positions de deux trajets distincts, comme c'est le cas pour l'approche présentée par Wang et Rushforth [Wang 1996]. Une restriction du voisinage, dictée par l'objectif de minimisation de la largeur de bande, y est effectuée en sélectionnant le premier trajet impliqué dans l'échange parmi ceux étant affecté à la plus grande fréquence ; le deuxième trajet est quant à lui sélectionné de façon aléatoire. Box [Box 1978] utilise la même stratégie à partir d'une liste générée de façon aléatoire. Le critère est ici de trouver une affectation réalisable utilisant un nombre fixe de fréquences consécutives : à partir de la liste, une affectation est générée. Si celle-ci s'avère irréalisable, les trajets dont le domaine est vide se voient affecter des poids qui vont servir à la génération de la liste suivante. Le processus est itéré jusqu'à obtention d'une solution réalisable ou atteinte d'un critère d'arrêt.

Des voisinage inspirés de ceux développés pour les problèmes de coloration de graphes peuvent également être rencontrés [Borgne 1994].

Finalement, Vasquez et al. [Vasquez 2003], puis Vlasaz et Vasquez [Vlasak 2003], proposent une méthode taboue plus originale basée sur la notion de voisinage consistant (cf. état de l'art)

pour résoudre, respectivement, un problème d'affectation avec polarisation et le problème proposé par le CELAR. Dans le cadre de ce dernier problème, il s'agit de résoudre une série de MAX-CSP (satisfaction du nombre maximal de contraintes de compatibilité électromagnétique) avec des largeurs de bande décroissantes.

### 3.2.4 Une méthode hybride ad-hoc

Une procédure présentant des similitudes avec l'approche que nous proposons existe pour le FAP. Celle-ci, dénommée "*solve and extend*", est basée sur une exécution en deux phases. Au cours de la première phase, un sous-problème "convenablement" difficile du problème original est sélectionné puis résolu. Il est ensuite étendu pendant la deuxième phase en vue d'obtenir une solution au problème global. Ces deux traitements sont itérés jusqu'à ce qu'une solution satisfaisante soit obtenue ou un critère d'arrêt atteint. Cette approche a été éprouvée par quelques auteurs parmi lesquels Smith et al. [Smith 1998] ou encore Mannino et Sassano [Mannino 2003] pour résoudre des problèmes de type MS-FAP. Les premiers génèrent un sous-problème initial en sélectionnant les trajets apparaissant dans une clique de niveau $p$ dans le graphe des contraintes puis résolvent le sous-problème induit de façon heuristique. L'extension à une solution complète est également réalisée de manière heuristique. Le processus est répété en ajoutant à chaque fois des sommets de saturation maximale à la clique initiale. La deuxième approche préconise quant à elle l'emploi d'une méthode d'énumération implicite complète pour la résolution des sous-problèmes successifs ainsi que leur extension à une assignation globale. Les sous-problèmes considérés sont obtenus les uns des autres en ajoutant des trajets suivant des critères de connectivité dans le graphe des contraintes.

## 3.3 Application de LSSPER au FAP

Nous allons au cours de cette section présenter la spécification de notre méthode de grands voisinages à la résolution du problème d'affectation proposé par le CELAR. Nous débutons en section 3.3.1 par l'exposé de deux techniques de pré-traitement appliquées en tout début de processus. Nous poursuivons par la présentation du schéma général d'exécution en section 3.3.2 puis détaillons les différentes caractéristiques de notre approche :

- la section 3.3.3 est consacrée à la description d'un algorithme glouton, qui est utilisé notamment pour générer la solution initiale ;
- les stratégies de sélection puis de résolution de sous-problèmes sont abordées en sections 3.3.4 et 3.3.5 ;
- la procédure d'obtention de la solution voisine est décrite en section 3.3.6 ;
- des caractéristiques auxiliaires de moindre importance sont présentées en section 3.3.7.

Nous exhibons finalement en section 3.3.8 les résultats expérimentaux obtenus sur les instances fournies par le CELAR.

### 3.3.1 Pré-traitement

L'objectif de cette phase préliminaire est de réduire la taille du problème en supprimant les valeurs inconsistantes des domaines des trajets et en étudiant la séparabilité du problème global. Nous avons pour cela procédé en deux phases distinctes. Cette partie s'avère nécessaire et profitable de par la nature de ces instances issues d'application réelles.

#### Réduction des domaines des trajets

Les contraintes impératives d'égalité sont utilisées pour supprimer immédiatement les valeurs inconsistantes des domaines des trajets en appliquant les règles suivantes :

$$\forall (i,j) \in E_{IE}, v \in D_i \mid v - \epsilon_{ij} \notin D_j, v + \epsilon_{ij} \notin D_j \quad \Rightarrow D_i = D_i - \{v\} \tag{3.19}$$

Cette réduction n'est effective que pour 14 des 30 instances et de façon efficace sur un nombre encore plus restreint, ainsi que l'on peut le constater en Table 3.2 : six instances seulement subissent des réductions importantes (autour de 20% de valeurs éliminées).

| Scénario FAPPG | Taille de l'union des domaines | Pourcentage de valeurs éliminées |
|---|---|---|
| 01_0016 | 10200 | 2.35% |
| 02_0018 | 11600 | 2.07% |
| 03_0066 | 39200 | 3.26% |
| 07_0182 | 108700 | 3.24% |
| 11_0164 | 97800 | 2.29% |
| 12_0902 | 540000 | 3.39% |
| 16_0038 | 3800 | 20% |
| 17_0040 | 4000 | 20% |
| 21_1088 | 533120 | 0.81% |
| 22_0768 | 299520 | 1.54% |
| 27_0154 | 9240 | 33.33% |
| 28_0398 | 99500 | 20% |
| 29_0526 | 210400 | 19% |
| 30_2166 | 866400 | 19% |

TAB. 3.2 – Réductions des domaines sur 14 instances

#### Décomposition du problème

Le deuxième point de pré-traitement consiste à identifier les différentes composantes connexes dans le graphe de contraintes $G = (T, E)$, en vue de pouvoir traiter indépendamment les uns des autres les problèmes réduits $P_1, \ldots, P_m$ qu'elles définissent. Ce traitement séparé est bien entendu équivalent au traitement du problème global uniquement dans le cadre du MI-FAP. En effet, lorsque le critère d'optimisation concerne la minimisation de la largeur de spectre, deux composantes

connexes distinctes peuvent jouer sur la valeur de cette dernière. Cependant, ce dernier cas de figure peut tout de même être traité en considérant des problèmes réduits indépendants grâce à l'utilisation de stratégies adaptées, ainsi que nous le verrons en section 3.3.4.

Ainsi que pour le problème global $P$, on peut définir pour chaque problème réduit $P_k$, les sous-graphes de contraintes associés. Soit $T_k$, l'ensemble des trajets appartenant au problème réduit $P_k$, on note $Gk_{IE} = (T_k, Ek_{IE})$ le sous-graphe de $G_{IE} = (T, E_{IE})$ restreint aux sommets de $T_k$. On définit de la même façon les sous-graphes $Gk_{ID} = (T_k, Ek_{ID})$, $Gk_C = (T_k, Ek_C)$ et $Gk_G = (T_k, Ek_G)$.

Le Tableau 3.3, qui répertorie pour chaque instance le nombre de *composantes connexes*, fait ressortir l'efficacité toute relative du pré-traitement avec uniquement 8 scénarios pouvant être décomposés correctement.

| Scénario FAPPG | Nombre de Composantes Connexes | Scénario FAPPG | Nombre de Composantes Connexes |
|---|---|---|---|
| 01_0016 | 1 | 16_0038 | 1 |
| 02_0018 | 1 | 17_0040 | 1 |
| 03_0066 | 2 | 18_0052 | 1 |
| 04_0064 | 2 | 19_0770 | 1 |
| 05_0064 | 1 | 20_1930 | 136 |
| 06_0182 | 4 | 21_1088 | 1 |
| 07_0182 | 4 | 22_0768 | 2 |
| 08_0608 | 20 | 23_0034 | 1 |
| 09_1460 | 65 | 24_0048 | 1 |
| 10_1698 | 73 | 25_0106 | 1 |
| 11_0164 | 3 | 26_0140 | 2 |
| 12_0902 | 23 | 27_0154 | 1 |
| 13_0306 | 2 | 28_0398 | 9 |
| 14_0194 | 1 | 29_0526 | 17 |
| 15_2454 | 4 | 30_2166 | 46 |

TAB. 3.3 – Nombre de composantes connexes sur les 30 instances du CELAR

### 3.3.2 Algorithme général

Pour chacune des deux phases de l'optimisation (minimisation des interférences puis de la largeur de bande), le même schéma global, décrit dans la figure 3.6, sera appliqué. Contrairement au schéma général donné au cours du chapitre 1, la particularité de l'algorithme est qu'il travaille ici sur une décomposition du problème général faisant intervenir les $m$ problèmes réduits $P_1, \ldots, P_m$ de taille respective $n_1, \ldots, n_m$ obtenus au terme de la phase de pré-traitement. Cette décomposition, de même que le pré-traitement de réduction des domaines, sont réalisés avant l'amorçage du processus de recherche locale. Ce dernier consiste en la génération puis la résolution, au cours de chaque itération $s$, de $m$ sous-problèmes distincts. Chaque sous-problème $SP_k^s$, de taille $p_k \leq n_k$, est généré à partir de la solution courante $\overline{F}^{s-1}$ et des caractéristiques de $P_k$ conformément à une stratégie

sous-jacente dictée par l'objectif courant. Il est ensuite résolu par une méthode de résolution adaptée qui délivre la solution $\overline{F}^s_k$. Le résultat de l'optimisation des $m$ sous-problèmes est alors utilisé pour générer la solution voisine $\overline{F}^s$, à partir de laquelle le processus est itéré. Ce dernier, qui retourne la solution finale $\overline{F}^*$, est répété jusqu'à l'atteinte d'un temps maximal d'exécution $MAX\_CPU$.

---

**Début**
1. pré-traitement
2. génération d'une solution initiale $\overline{F}^0$ à l'aide d'un algorithme glouton
3. $\overline{F}^* := \overline{F}^0$
4. $s := 1$
5. **Répéter**
6. **Pour chaque** problème réduit $P_k$
7. génération du sous-problème $SP^s_k$ à partir de $\overline{F}^{s-1}$, $P_k$ et de l'objectif courant
8. Résolution du sous-problème : $\overline{F}^s_k := solve(SP^s_k)$
9. **Fin Pour**
10. Génération de la solution voisine $\overline{F}^s$
11. **Si** $f(\overline{F}^s) < f(\overline{F}^*)$ **Alors**
12. $\overline{F}^* := \overline{F}^s$
13. **Fin Si**
14. $s := s + 1$
15. **Jusqu'à** $nb\_CPU := MAX\_CPU$

**Fin**

---

FIG. 3.6 – *Algorithme général de LSSPER pour le FAP*

### 3.3.3 Génération de la solution initiale

Dans cette partie, nous allons expliquer le principe de fonctionnement d'un algorithme glouton utilisé pour la génération de la solution initiale.

Le processus consiste à choisir à chaque itération le sommet de plus petit domaine possédant le degré maximal dans le graphe des contraintes et à l'affecter à la fréquence la plus petite possible parmi celles qui minimisent la somme pondérée des contraintes co-site et globales violées. Les contraintes impératives nécessitant d'être absolument respectées, l'affectation d'un trajet à une fréquence de son domaine implique des réductions de domaine sur les trajets liés à celui-ci par le biais de telles contraintes. Ainsi, lorsque le domaine d'un trajet devient vide, l'algorithme n'assure pas l'obtention d'une solution réalisable. Pour éviter de générer une solution initiale non réalisable, une procédure arborescente classique est initiée en cas d'échec de l'algorithme afin de rechercher une solution satisfaisant uniquement les contraintes impératives. Le schéma de branchement invoqué au cours de cette procédure est identique à celui du glouton. Bien qu'un tel cas de figure ne se soit jamais présenté, la procédure à toutefois été éprouvée sur chacune des instances afin de valider son utilisation hypothétique : il s'est avéré qu'en raison de la taille importante des domaines et du nombre relativement restreint de contraintes impératives celle-ci était d'une grande rapidité d'exécution et donc tout à fait à même de palier une déficience de l'algorithme glouton.

### 3.3.4 Génération d'un sous-problème

À l'itération courante $s$, à partir de la solution courante $\overline{F}^{s-1}$, un sous-problème $SP_k^s$ est défini pour chaque problème réduit $P_k$ en libérant un sous-ensemble $T_k^s = \{t_1^s, \ldots, t_{p_k}^s\} \subseteq T_k$ de $p_k$ trajets. Les autres trajets demeurant fixés à la fréquence qui leur est allouée dans la solution courante, le sous-problème consiste alors à rechercher une solution sous certaines contraintes. Ainsi, les trajets fixes imposent des réductions sur les domaines des trajets libres par le biais des contraintes impératives et, éventuellement, des contraintes co-site. De même des réductions additionnelles, basées sur l'étude de la solution courante et relatives à des stratégies de résolution, sont réalisées lorsque le critère de minimisation concerne la largeur de bande. Les contraintes résiduelles du problème doivent ensuite être spécifiées ainsi que l'objectif à optimiser. Dans le cadre de la minimisation des interférences, lorsqu'aucune contrainte de $P_k$ n'est violée, le sous-problème $SP_k^s$ n'est pas généré.

#### Sélection des trajets

Plusieurs stratégies de constitution de l'ensemble $T_k^s$ ont été éprouvées. Elles débutent toutes de la même façon : un trajet initial $t_1^s$, appartenant à une contrainte co-site ou globale violée dans le cas de la minimisation des interférences, est choisi de façon aléatoire. $T_k^s$ est ensuite étendu selon l'une des méthodes suivantes :

- Aléatoire : les trajets restants sont sélectionnés de façon aléatoire.
- Impérative : on considère le sous-graphe de contraintes $Gk_I = (T_k, Ek_I), Ek_I = Ek_{IE} \cup Ek_{ID}$ restreint aux trajets de $T_k$. L'ensemble des trajets correspondants aux sommets adjacents du sommet $t_1^s$ est inséré dans $T_k^s$, dans la limite de $p_k$ trajets. Si cette limite n'est pas atteinte alors le processus est itéré à partir du deuxième, puis troisième,..., trajet ajouté à l'ensemble $T_k^s$. S'il n'est plus possible à un moment donné de rajouter des trajets à partir de ceux précédemment inclus, alors le processus est relancé à partir d'un nouveau trajet sélectionné de façon aléatoire.
- Co-site : le processus est le même que précédemment à l'exception du sous-graphe de contraintes considéré. Ici, c'est le sous-graphe $Gk_C = (T_k, Ek_C)$ qui sert de support à l'inclusion des trajets ultérieurs.
- Globale : le sous-graphe de contraintes $Gk_G = (T_k, Ck_G)$ défini par les contraintes globales est utilisé pour étendre l'ensemble $T_k^s$. Les sommets associés aux trajets apparaissant dans une même contrainte globale définissent une clique du graphe.
- Toutes_Contraintes : le graphe de contraintes $Gk = (T_k, Ek)$ complet sert à l'élaboration du sous-problème. Ce processus est illustré en Figure 3.7 sur l'instance FAPPG02_0018. A partir du trajet 2, sélectionné de façon aléatoire parmi l'ensemble des trajets impliqués dans une contrainte électromagnétique violée, l'ensemble $T_1^s$ de taille $p_1 = 10$ est constitué. En partie (a) de la figure, les trajets adjacents au trajet 2 dans le graphe de contraintes sont ajoutés à $T_1^s$. Puis le processus est itéré à partir du trajet 10, deuxième trajet inclus dans $T_1^s$ : aucune adjonction n'est ici réalisée car tous les trajets adjacents au trajet 10 appartiennent déjà à $T_1^s$. Finalement, en partie (c), $T_1^s$ est complété par le trajet 5 qui est relié au trajet 8, troisième trajet inclus dans $T_1^s$. Pour plus de lisibilité, chaque partie de la figure représente uniquement le sous-graphe de contraintes issu du trajet $i$ : $Gk_i = (T_k, Ek_i), Ek_i = \{(i,j)|(i,j) \in Ek\}$.

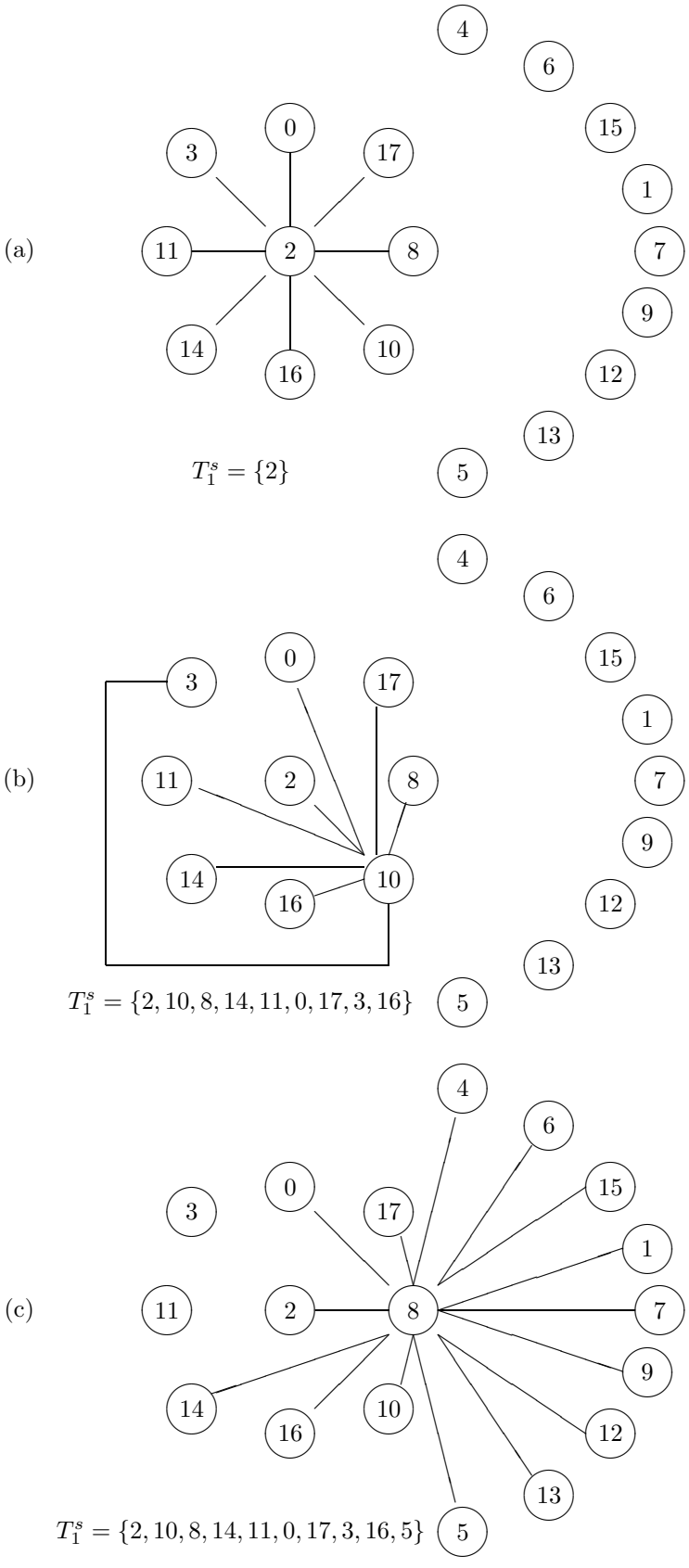


FIG. 3.7 – *Exemple de sélection d'un sous-problème*

**Réduction de domaines**

La fixation de certains trajets à leur valeur courante induit des réductions de domaines pour les autres trajets par le biais des contraintes impératives d'égalité et de différence. Dans le cadre de la minimisation de la largeur de bande, le respect des contraintes co-site induit également la suppression de nombre de valeurs inconsistantes des domaines des trajets libérés. Les réductions de domaines réalisées sont les suivantes :

– réduction par contraintes impératives d'égalité :

$$\forall (i,j) \in Ek_{IE}, j \in T_k - T_k^s, i \in T_k^s, D_i = D_i \cap \{\overline{f}_j - \epsilon_{ij}, \overline{f}_j + \epsilon_{ij}\} \tag{3.20}$$

– réduction par contrainte impératives de différence :

$$\forall (i,j) \in Ek_{ID}, j \in T_k - T_k^s, i \in T_k^s, D_i = D_i - \{\overline{f}_j - \epsilon_{ij}\} - \{\overline{f}_j + \epsilon_{ij}\} \tag{3.21}$$

– réduction par contraintes co-site :

$$\forall (i,j) \in Ek_C, j \in T_k - T_k^s, i \in T_k^s, D_i = D_i - ]\overline{f}_j - \delta_{ij}, \overline{f}_j + \delta_{ij}[ \tag{3.22}$$

Nous prenons en outre le parti de réaliser des réductions additionnelles liées à des stratégies de résolution. Celles-ci ne correspondent donc plus forcément à des conditions nécessaires de réalisabilité mais à des suppression heuristiques de valeurs. Soient $f_+ = \max_{i \in T} \overline{f}_i^{s-1}$ et $f_- = \min_{i \in T} \overline{f}_i^{s-1}$, les valeurs de fréquences maximale et minimale affectées dans la solution courante $\overline{F}^{s-1}$, on détermine alors les stratégies suivantes :

– stratégie AllMax : elle est invoquée lorsque tous les trajets assignés à $f_+$ appartiennent à l'ensemble $T_k^s$ et au moins un trajet assigné à $f_-$ n'y appartient pas. La valeur de plus petite fréquence utilisée étant fixée, une façon d'obtenir une solution de span inférieur consiste à affecter aux trajets du sous-problème des fréquences strictement inférieures à $f_+$ et supérieures à $f_-$. Les domaines subissent ainsi la réduction suivante :

$$\forall i \in T_k^s, D_i = D_i \cap [f_-, f_+[ \tag{3.23}$$

– stratégie AllMin : elle constitue le pendant de la stratégie précédente et s'exécute lorsque tous les trajets affectés à $f_-$ appartiennent au sous-problème et au moins un trajet demeure fixé à $f_+$. Les domaines sont ajustés de la façon suivante :

$$\forall i \in T_k^s, D_i = D_i \cap ]f_- + 1, f_+] \tag{3.24}$$

– stratégie NoMinMax : elle intervient lorsqu'au moins un trajet de $T - T_k^s$ reste affecté à $f_-$ et un autre à $f_+$. La valeur de span globale ne peut ici pas être améliorée mais un réarrangement des fréquences affectées à l'intérieur de l'intervalle $[f_-, f_+]$ peut néanmoins conduire à une amélioration au cours d'une itération ultérieure. Les domaines fréquentiels subissent ainsi la

réduction suivante :

$$\forall i \in T_k^s, D_i = D_i \cap [f_-, f_+] \tag{3.25}$$

– stratégie AllMinMax : elle est utilisée lorsque tous les trajets affectés à $f_+$ et $f_-$ appartiennent à l'ensemble $T_k^s$. Soient les distances $d_- = \min_{i \notin T_k^s} \overline{f}_i^{s-1} - f_-$ et $d_+ = f_+ - \max_{i \notin T_k^s} \overline{f}_i^{s-1}$. Toute solution améliorant la valeur de span ne possédera pas de trajets affectés à des fréquences hors de l'intervalle $[f_- - d_+, f_+ + d_-]$. Les domaines sont donc réduits comme suit :

$$\forall i \in T_k^s, D_i = D_i \cap [f_- - d_+, f_+ + d_-] \tag{3.26}$$

L'emploi de ces stratégies permet de faciliter la résolution du sous-problème, les affectations des fréquences aux trajets étant davantage contraintes. Si elles entraînent possiblement la suppression de solutions de bonne qualité, voire optimales, du voisinage, elles permettent cependant de résoudre des sous-problèmes de taille plus importante. En outre, leur diversité ainsi que le schéma de sélection non uniforme du sous-problème (cf. section 3.3.7) permet d'espérer considérer dans la suite du processus des solutions de bonne qualité écartées à une itération donnée par l'une de ces stratégies.

**Contraintes**

$$f_i \in D_i \qquad \forall i \in T_k^s \tag{3.27}$$
$$f_i = \overline{f}_i^{s-1} \qquad \forall i \in T_k - T_k^s \tag{3.28}$$
$$|f_i - f_j| = \epsilon_{ij} \qquad \forall\ (i,j) \in Ek_{IE}, i,j \in T_k^s \tag{3.29}$$
$$|f_i - f_j| \neq \epsilon_{ij} \qquad \forall\ (i,j) \in Ek_{ID}, i,j \in T_k^s \tag{3.30}$$
$$|f_i - f_j| \geq \delta_{ij} \qquad \forall\ (i,j) \in Ek_C, \{i,j\} \cap T_k^s \neq \emptyset \tag{3.31}$$
$$\textstyle\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(i)|f_i - f_j|} \leq \Lambda_i \quad \forall\ C_i \in Ck_G, (P_i \cup \{i\}) \cap T_k^s \neq \emptyset \tag{3.32}$$

Le respect strict des contraintes de compatibilité électromagnétique n'est bien sûr indispensable que pour la minimisation de la largeur de bande. Dans l'autre cas, on définit les ensembles $V_k^s$ et $W_k^s$ comme suit :

$$V_k^s = \{|f_i^s - f_j^s| < \delta_{ij}, (i,j) \in Ek_C\} \tag{3.33}$$
$$W_k^s = \{\textstyle\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)|f_i^s - f_j^s|} > \Lambda_i, C_i \in Ck_G\} \tag{3.34}$$

**Objectifs**

Il s'agit à présent de spécifier l'objectif du sous-problème. Celui-ci dépend bien évidemment du critère à optimiser. Pour la minimisation de la largeur de bande, le sous-problème consiste soit à trouver une solution réalisable satisfaisant l'ensemble des contraintes (stratégies AllMax, AllMin,

NoMinMax), soit à optimiser la fonction objectif suivante (stratégie AllMinMax) :

$$\min(\max_{i \in T_k^s} f_i - \min_{i \in T_k^s} f_i) \tag{3.35}$$

Concernant la minimisation des interférences, il s'agit de trouver une solution qui améliore la valeur de fonction objectif, à savoir :

$$\alpha|V_k^s| + \beta|W_k^s| < \alpha|V_k^{s-1}| + \beta|W_k^{s-1}| \tag{3.36}$$

avec $V_k^{s-1} = \{|\overline{f}_i^{s-1} - \overline{f}_j^{s-1}| < \delta_{ij}, (i,j) \in Ek_C\}$
et $W_k^{s-1} = \{\sum_{j \in P_i} \lambda_{ij} T_{\sigma(i)\sigma(j)|\overline{f}_i^{s-1} - \overline{f}_j^{s-1}|} > \Lambda_i, C_i \in Ck_G\}$

### 3.3.5 Résolution d'un sous-problème

La résolution d'un sous-problème $SP_k^s$ est réalisée en considérant deux critères : d'une part, l'objectif à satisfaire, d'une autre la stratégie de résolution de haut niveau elle-même. Concernant le critère d'optimisation, il est important de souligner qu'en raison des stratégies et des divers objectifs qu'il met en place, la résolution d'un sous-problème ne conduit pas nécessairement à l'obtention d'une solution optimale pour un critère donné (ce n'est d'ailleurs le cas que pour des sous-problèmes bien particuliers), ni même améliorante (stratégie NoMinMax lors de la minimisation de la largeur de bande).

La stratégie de résolution de haut niveau détermine quant à elle l'effort calculatoire qui va être réalisé pour trouver une solution $\overline{F}_k^s$ au sous-problème $SP_k^s$ : résolution heuristique rapide (procurant des solutions de qualité moyenne) ou résolution exacte plus gourmande (procurant des solutions potentiellement de meilleure qualité), chaque approche étant couplée à des procédés de propagation de contraintes.

La procédure heuristique utilisée est l'algorithme glouton présenté en section 3.3.3. Comme indiqué précédemment, cette méthode n'assure en aucun cas l'obtention d'une solution améliorante, auquel cas la solution courante demeure inchangée : $\overline{f}_i^s = \overline{f}_i^{s-1} \forall i \in T_k$. Cependant, elle se comporte globalement de façon efficace lorsqu'il s'agit d'améliorer une solution de qualité moyenne. C'est pourquoi elle est employée au début du processus global, lors de la phase de minimisation des interférences, afin de permettre de trouver rapidement une solution courante de qualité satisfaisante.

Une deuxième politique de résolution est alors instaurée. Celle-ci repose sur l'emploi d'une méthode arborescente tronquée. Le schéma de branchement consiste à choisir le trajet de plus petit domaine et à l'affecter à la fréquence la plus petite possible (parmi celles qui minimisent la somme pondérée des contraintes violées lorsque l'objectif concerne la minimisation de l'interférence). La recherche est stoppée dès qu'une solution réalisable ou optimale (first-fit), selon les cas, est atteinte, ou lorsqu'un temps d'exécution maximal $H$ est dispensé, auquel cas la solution courante demeure inchangée. Cette méthode permet d'explorer de façon plus approfondie le voisinage défini par un sous-problème et de fait d'intensifier la procédure de recherche. Le temps de résolution maximal alloué dépend du type de sous-problème : il est élevé pour la recherche d'une solution améliorant la valeur d'interférence ou lorsque la stratégie AllMinMax est active ; au contraire, il est très bref pour les autres cas de minimisation de la largeur de bande, en particulier lorsque la stratégie NoMinMax

est employée (simple réagencement de fréquences à l'intérieur d'une portion de spectre donnée).

### 3.3.6 Construction de la solution voisine

Au cours d'une itération, la résolution des $m$ sous-problèmes successifs détermine la solution voisine $\overline{F}^s = (\overline{F}^s_1, \ldots, \overline{F}^s_m)^t$ de $\overline{F}^{s-1}$. Aucune procédure particulière de post-optimisation n'est ici mise en place, contrairement à ce qui a été réalisé pour le RCPSP. Il pourrait cependant être intéressant de concevoir une telle procédure qui pourrait par exemple consister à résoudre le problème global en limitant pour chaque trajet une plage de fréquences d'affectation restreinte autour de la valeur courante.

### 3.3.7 Autres caractéristiques

#### Ajustement de la valeur de $p_k$

La taille $p_k$ d'un sous-problème est dépendante de la méthode de résolution employée. Ainsi, $p_k = p_{exact}$ lorsqu'une procédure arborescente est invoquée pour la résolution de $SP^s_k$, $p_k = p_{heur}$ dans le cas inverse, avec $p_{heur} \gg p_{exact}$. Ainsi que pour le RCPSP, une tentative d'auto-ajustement a été réalisée, celle-ci ne se révélant pas effective dans le cadre de l'utilisation de la méthode exacte. Nous ne faisons ainsi varier la valeur de $p_k$ que lors de la phase de recherche faisant intervenir l'utilisation de l'algorithme glouton. Le temps d'exécution négligeable de cette heuristique ne constituant pas un bon critère d'ajustement, le paramètre invoqué concerne la stagnation de la solution courante au niveau d'un sous-optimum local. Ainsi, lorsque la solution associée à un sous-problème $P_k$ donné n'a pas été améliorée durant un nombre donné d'itérations, la valeur de $p_k$ est modifiée (incrémentée ou réinitialisée) afin d'espérer échapper au piège des optima locaux.

#### Diversification

Ainsi que pour le RCPSP, une composante aléatoire est insérée dans le processus de sélection des trajets afin de permettre une certaine diversification dans la considération du voisinage. Ainsi, les trajets seront considérés pour leur inclusion dans $T^s_k$ dans l'ordre donné par une permutation aléatoire des indices. De même que précédemment, ce biais offre la possibilité d'obtenir à partir d'une même solution courante des sous-problèmes potentiellement très différents les uns des autres et ainsi de considérer une portion de l'espace de recherche inédite.

### 3.3.8 Résultats expérimentaux

Nous avons développé la méthode proposée en C++ conjointement avec l'utilisation des outils ILOG (Solver et Concert). Nous avons fixé de façon empirique le temps maximal de résolution de la méthode exacte à 30 secondes tandis que le temps maximal d'exécution de l'algorithme était défini par les termes de l'appel d'offre à une heure.

Les expérimentations, dont les résultats sont présentés dans le Tableau 3.4, ont été conduites au sein du CELAR sur un PC équipé d'un Pentium II cadencé à 350 MHz et possédant de 256 Mo

de RAM. Afin de pouvoir établir une comparaison, nous avons confronté les résultats obtenus par LSSPER à ceux obtenus par les autres équipes travaillant sur l'appel d'offre : Vlasak et Vasquez [Vlasak 2003] et leur méthode taboue à voisinage consistant (TS), Sarzeaud [Sarzeaud 2003] et sa procédure de recuit simulé (SA) (cf. section 3.2.3 pour une description de ces méthodes). Afin que la comparaison soit équitable, l'ensemble des tests ont été réalisés selon le même mode opératoire (même limite de temps, même matériel). Nous avons reporté pour chacune des approches les résultats obtenus en terme d'interférence et de valeur de largeur de bande : lorsque la solution n'est pas réalisable en regard des contraintes de compatibilité électromagnétique, la valeur d'interférences, ainsi que la largeur de bande associée au problème (entre parenthèses), sont reportées en colonne (MI-FAP) ; au contraire, lorsque la solution satisfait l'ensemble des contraintes, la valeur de largeur de bande est reportée en colonne (MS-FAP). Les meilleures solutions sont indiquées en gras. Une étoile indique l'optimalité de la solution en regard de l'objectif considéré (voir [Oliva 2003] et [Palpant 2003b] pour des études sur le sujet).

| Scénario | LSSPER | | SA [Sarzeaud 2003] | | TS [Vlasak 2003] | |
|---|---|---|---|---|---|---|
| FAPPG | MI-FAP | MS-FAP | MI-FAP | MS-FAP | MI-FAP | MS-FAP |
| 01_0016 | - | **548*** | - | 549 | - | **548*** |
| 02_0018 | - | **629** | - | **629** | - | **629** |
| 03_0066 | 2 (599) | - | **2** (**580**) | - | 2 (623) | - |
| 04_0064 | - | 520 | - | 520 | - | **519** |
| 05_0064 | - | **599** | - | 623 | - | 676 |
| 06_0182 | - | **718** | - | **718** | - | 758 |
| 07_0182 | 6 (666) | - | **4** (**698**) | - | 8 (687) | - |
| 08_0608 | - | **620** | - | 646 | - | 642 |
| 09_1460 | - | **544** | - | 656 | - | 860 |
| 10_1698 | - | **412** | - | 692 | - | 849 |
| 11_0164 | - | 604 | - | 656 | - | **601** |
| 12_0902 | - | **572** | 1 (639) | - | 2 (634) | - |
| 13_0306 | 9 (399) | - | **6** (**399**) | - | 8 (380) | - |
| 14_0194 | - | 398 | - | 360 | - | **354** |
| 15_2454 | **31** (**399**) | - | 73 (399) | - | 44 (399) | - |
| 16_0038 | **46** (**146**) | - | **46** (**146**) | - | **46** (**146**) | - |
| 17_0040 | 46 (99) | - | **45** (**98**) | - | **45** (**98**) | - |
| 18_0052 | - | **404** | 1 (476) | - | - | 408 |
| 19_0770 | 4385 (492) | - | 4375 (496) | - | **3998** (**496**) | - |
| 20_1930 | **150** (**492**) | - | 193 (496) | - | 152 (496) | - |
| 21_1088 | - | **982** | 2 (964) | - | 4 (994) | - |
| 22_0768 | - | **788** | - | 818 | 1 (894) | - |
| 23_0034 | - | **380*** | - | **380*** | - | **380*** |
| 24_0048 | - | **410*** | - | 430 | - | **410*** |
| 25_0106 | 2* (540) | - | **2*** (**490**) | - | 2* (540) | - |
| 26_0140 | - | **480** | 1 (492) | - | - | **480** |
| 27_0154 | **2*** (**490**) | - | 4 (490) | - | **2*** (**490**) | - |
| 28_0398 | - | **610** | - | 646 | - | 638 |
| 29_0526 | - | **542** | - | 852 | - | 866 |
| 30_2166 | **23** (**912**) | - | 27 (912) | - | 32 (912) | - |

TAB. 3.4 – Résultats expérimentaux sur les 30 instances du CELAR

Les résultats apparaissent compétitifs, quel que soit le critère considéré. Concernant l'objectif de minimisation des interférences, 19 instances ont été trouvées réalisables, dont 8 en un temps très court (pas plus de 5 minutes), et deux instances supplémentaires sont optimales (FAPPG25_0106 et FAPPG27_0154). Certaines instances plus difficiles sont de plus bien résolues. Ce bon comportement est particulièrement visible sur l'instance FAPPG15_2454, de très grande taille, avec seulement 31 contraintes violées. Concernant le critère de largeur de bande, trois instances sont optimales. Si l'on se situe à un niveau plus général, la comparaison avec les méthodes TS ou SA souligne encore le bon comportement de LSSPER, avec 21 meilleures solutions obtenues contre 12 pour la méthode taboue et 9 pour la procédure de recuit simulé. Sur l'instance FAPPG10_1698, la valeur de largeur de bande améliore de façon spectaculaire celle obtenue par les autres approches. De façon générale, un des aspects les plus frappants révélé par la confrontation des résultats est l'efficacité indéniable de l'approche en terme de largeur de bande obtenue sur les instances hautement décomposables. En effet, sur la totalité de ces instances les résultats surclassent littéralement ceux obtenus par les autres méthodes. Il semble donc que les stratégies de résolution heuristique se comportent de la meilleure façon lorsqu'elles sont appliquées sur ce type d'instances.

Il s'agit toutefois de souligner la contre-performance réalisée sur l'instance FAPPG19_0770 pour laquelle la valeur globale d'interférence est plus élevée. Cette instance étant la seule faisant intervenir des coefficients de pondération différents pour les contraintes globales et co-site, on peut raisonnablement penser que la stratégie de sélection des sous-problèmes est moins adaptée à ce cas particulier, ce qui souligne encore davantage, s'il en était besoin, l'importance fondamentale que revêt le choix des sous-problèmes à optimiser.

Afin de vérifier ces propos, nous avons reporté dans le Tableau 3.5 les résultats obtenus par les différentes stratégies de sélection des sous-problèmes lorsque le processus de résolution est itéré durant un nombre fixé d'itérations. Celui-ci dépeint les résultats moyens obtenus sur l'ensemble des instances en terme de qualité de solution (interférences et largeur de bande en colonnes 2 et 3 respectivement) aussi bien qu'en terme de temps d'obtention de la meilleure solution.

| stratégie | interférence | largeur | temps CPU |
|---|---|---|---|
| Aléatoire | 206.97 | 605.2 | 4039.67 |
| Impérative | 92.43 | 622.8 | 4950.43 |
| Co-site | 167.4 | 618.87 | 3438.67 |
| Globale | 175.6 | 612.1 | 3370.1 |
| Toutes_Contraintes | 120.73 | 614.17 | 3132.53 |

TAB. 3.5 – Comparaison des stratégies de sélection

Ces résultats révèlent une certaine homogénéité des différentes stratégies en ce qui concerne le critère de largeur de bande. Ils font également ressortir sans ambiguïté l'écrasante dominance de la stratégie "Impérative", qui réalise la sélection par l'intermédiaire des contraintes impératives, lorsque l'on se réfère au critère de minimisation des interférences. Dans une moindre mesure, la stratégie "Toutes_Contraintes", qui sélectionne des trajets reliés par un quelconque type de contraintes, se comporte également de façon satisfaisante. En outre, en raison de temps d'exécution bien moindres qu'elle obtient, nous avons porté notre choix sur cette dernière stratégie de sélection.

Nous avons également testé notre approche sur les instances homologues (FAPPI) dans les-

quelles les contraintes globales sont remplacées par des ensembles de contraintes co-site beaucoup plus restrictives, selon le processus de répartition uniforme du "droit à perturber" décrit dans la section 3.1.3. L'objectif avoué est de démontrer l'impact des contraintes globales sur la qualité des solutions. Les résultats obtenus sont reportés dans le tableau 3.6. Lorsqu'une solution réalisable en regard des contraintes de compatibilité électromagnétique est obtenue, la largeur de bande obtenue est reportée en colonne MS-FAP. A l'inverse, lorsque la solution retournée n'est pas admissible, la valeur d'interférences est reporté en colonne MI-FAP. Dans les deux cas, les valeurs entre parenthèses indiquent les résultats obtenus sur le jeu d'instances originel.

| Scénario FAPPI | MI-FAP | MS-FAP |
|---|---|---|
| 01_0016 | - | 571 (548) |
| 02_0018 | 1 (0) | - |
| 03_0066 | 3 (2) | - |
| 04_0064 | - | 718 (520) |
| 05_0064 | 1 (0) | - |
| 06_0182 | - | 761 (718) |
| 07_0182 | 17 (6) | - |
| 08_0608 | - | 620 (620) |
| 09_1460 | - | 650 (544) |
| 10_1698 | - | 500 (412) |
| 11_0164 | - | 617 (604) |
| 12_0902 | - | 600 (572) |
| 13_0306 | 13 (9) | - |
| 14_0194 | 1 (0) | - |
| 15_2454 | 31 (31) | - |
| 16_0038 | 57 (46) | - |
| 17_0040 | 55 (46) | - |
| 18_0052 | - | 488 (404) |
| 19_0770 | 5629 (4385) | - |
| 20_1930 | 178 (150) | - |
| 21_1088 | 24 (0) | - |
| 22_0768 | 10 (0) | - |
| 23_0034 | - | 380 (380) |
| 24_0048 | - | 440 (410) |
| 25_0106 | 2 (2) | - |
| 26_0140 | - | 480 (480) |
| 27_0154 | 2 (2) | - |
| 28_0398 | - | 634 (610) |
| 29_0526 | - | 548 (542) |
| 30_2166 | 27 (23) | - |

TAB. 3.6 – Etude comparative du gain induit par les contraintes globales

Les résultats font ressortir de façon très nette la diminution importante du nombre d'instances libres d'interférences (13 contre 19 pour l'approche avec contraintes globales) tandis qu'aucune instance réalisable n'améliore la valeur de span de l'instance FAPPG correspondante. Tout concorde à démontrer l'apport réel de la prise en compte simultanée des interférences modélisée par les

contraintes globales.

Cependant, ces contraintes globales sont plus difficiles à prendre en compte que les contraintes binaires classiques. Les techniques de propagation notamment sont peu puissantes. Bien que le problème soit moins contraint il est alors, paradoxalement, plus difficile à résoudre. Ceci est particulièrement visible pour les sous-problèmes, qui sont mal résolus par une approche exacte dès lors que la taille augmente sensiblement. En raison de ces contraintes donc, et de la grande taille de certaines des instances, des adaptations nécessaires, et essentielles, ont été intégrées au sein de LSSPER :

- choix d'un algorithme glouton performant comme outil de résolution des sous-problèmes en début de processus, ce qui a permis notamment la résolution de sous-problèmes de taille plus importante ;
- réductions heuristiques visant à diminuer la complexité du voisinage réalisées en relation avec l'objectif à optimiser ;
- recherche de l'optimum très ponctuelle, associée à une méthode de résolution plus complète (utilisation d'une procédure arborescente tronquée au bout d'un temps plus important d'exécution).

Malgré une plus grande lourdeur de mise en place, l'ensemble de ces techniques a permis d'obtenir une procédure dont l'efficacité transparaît de par les résultats obtenus sur les jeux d'instances du CELAR. L'analyse de ces derniers a de plus permis de mettre en lumière certains points clés dans le processus général.

Il apparaît ainsi que les techniques de réduction invoquées lors de la minimisation de la largeur de bande sont particulièrement efficaces sur les instances hautement décomposables, ce qui laisse la place à une nouvelle piste d'étude. Il pourrait en effet être intéressant de mettre au point une technique de décomposition systématique pouvant être appliquée à l'ensemble des instances afin de pouvoir y appliquer les techniques sus-citées.

Un autre résultat frappant concerne la contre-performance de l'approche sur l'unique instance faisant intervenir des pondérations différentes dans le calcul de la fonction de coût. Il semblerait donc que les sous-problèmes générés soient moins adaptés à ce cas particulier. On pourrait ainsi s'interroger sur d'autres mécanismes de sélection des trajets libérés, par exemple la pondération de la sélection des trajets par les coûts associés aux diverses contraintes dans lesquelles ils apparaissent.

# Chapitre 4

# Une méthode incomplète basée sur Resolution Search et la PPC pour la résolution du problème des Queens_$n^2$

*Nous clôturons cette thèse par la présentation d'une méthode hybride basée sur la méthodologie de Resolution Search. Cette dernière consiste en une recherche complète originale que nous présenterons en section 4.1. Celle-ci se présente comme une méthode de recherche sur des configurations partielles dont les mouvements diffèrent de ceux employés dans les méthodes présentées au paragraphe 1.3. Nous proposons d'intégrer au sein de cette approche des composants heuristiques en vue de l'appliquer à la résolution du problème des Queens_$n^2$. Cette hybridation s'inscrit dans le cadre de la première catégorie de techniques présentées au paragraphe 1.2, à savoir les méthodes exactes incomplètes. Nous commencerons par présenter le problème des Queens_$n^2$ en section 4.2, qui constitue un cas particulier des problèmes de coloration de graphes (ou GCP pour Graphs Colouring Problem). Nous détaillerons ensuite l'application d'une procédure combinant Resolution Search "pure" avec des techniques de PPC sur ce problème particulier avant de terminer par l'hybridation retenue. Celle-ci repose sur des caractéristiques bien particulières des instances. Des résultats expérimentaux préliminaires obtenus sur un certain nombre d'instances seront finalement exposés. S'ils se situent qualitativement assez loin derrière les meilleurs résultats de la littérature, ils demeurent toutefois encourageants. Aussi ce chapitre se veut-il avant tout prospectif et tend à ouvrir des pistes pour l'orientation future des travaux.*

## 4.1 Resolution search

En 1997, Vašek Chvátal présente une méthode exhaustive pour la résolution de problèmes linéaires en variables binaires qu'il nomme Resolution Search [Chvátal 1997]. Cette dernière se démarque des procédures arborescentes classiques par une exploration originale de l'espace de recherche, censée permettre de rendre la résolution du problème moins largement dépendante de la stratégie de branchement invoquée. Celle-ci, à l'instar des procédés de backtracking intelligents, est basée sur le principe d'*apprentissage* qui consiste à *identifier* et *mémoriser* les raisons des échecs qui surviennent tout au long de la recherche. Lorsqu'un échec est détecté au cours de l'exploration l'espace de recherche, il s'agit alors, à partir de la configuration partielle courante, de déterminer l'ensemble des variables dont l'instanciation est responsable de l'échec. Interdire pour la suite de la recherche ladite instanciation partielle revient à générer une contrainte additionnelle, appelée *nogood*, au problème afin de couper la recherche en amont du nœud considéré. Resolution Search propose un schéma élégant de gestion de ces nogoods. Celui-ci permet non seulement de limiter l'espace mémoire et le temps de traitement des nogoods, mais aussi d'indiquer rapidement comment poursuivre la recherche à la suite d'un échec, tout en assurant la convergence de l'algorithme.

Nous introduisons en section 4.1.1 les notations et définitions relatives à ce principe et qui seront utilisées tout au long de ce chapitre puis poursuivons par la présentation détaillée des points-clé de l'approche.

### 4.1.1 Notations

#### Généralités

Soit le problème d'optimisation combinatoire en variables binaires exprimé comme suit :

$$P \ : \ \min f(X), X = (x_1, \ldots, x_n), AX \geq b, X \in \{0,1\}^n \tag{4.1}$$

Une instanciation partielle des variables de ce problème est donnée par un vecteur

$$u = (u_1, \ldots, u_n) \in \{0, 1, *\}^n \tag{4.2}$$

où $u_i$ correspond à l'instanciation de la variable $x_i$ avec $u_i = *$ si $x_i$ est non affectée. À chaque instanciation partielle $u$, on peut faire correspondre une valeur $oracle(u_1, \ldots, u_n)$ équivalente à la valeur optimale du problème linéaire relâché :

$$\min f(X), X = (x_1, \ldots, x_n), AX \geq b, X \in [0,1]^n, x_i = u_i \ \forall \ u_i \neq * \tag{4.3}$$

Soient $u$ et $v$, deux instanciations partielles, telles que $v_i = u_i \ \forall u_i \neq *$. La notion de *sous-instanciation*, définissant une relation d'*ordre partiel* sur $\{0, 1, *\}^n$, notée $\sqsubseteq$, stipule alors que $u$ est

une sous-instanciation de $v$, ou encore que $v$ est une *extension* de $u$ (l'espace de recherche associé à $v$ est contenu dans celui associé à $u$) :

$$(u_1, \ldots, u_n) \sqsubseteq (v_1, \ldots, v_n) \tag{4.4}$$

De façon triviale, *oracle* est une *fonction non-décroissante* sur $(\{0, 1, *\}^n, \sqsubseteq)$ ($oracle(u) \leq oracle(v)\ \forall\ u \sqsubseteq v$).

**Clauses**

Resolution Search s'attache à considérer des instanciations partielles $u$ dont l'espace associé est supprimé de la recherche. $u$ peut ainsi être considérée comme une *clause* dans le sens où toutes les solutions (partielles) ultérieures doivent vérifier la propriété suivante :

$$\bigvee_{i|u_i \neq *} x_i \neq u_i \tag{4.5}$$

Afin de simplifier les notations, nous exprimerons $u$ comme une clause sous *forme normale disjonctive* :

$$\bigvee_{i \in I_0} x_i \vee \bigvee_{i \in I_1} \overline{x}_i \tag{4.6}$$

où les symboles $x_i$ et $\overline{x}_i$, appelés *littéraux*, correspondent respectivement aux indices $i \in I_0$ tels que $u_i = 0$ et $i \in I_1$ tels que $u_i = 1$.

$u$ peut également être exprimée comme un ensemble de littéraux :

$$\{x_i | i \in I_0\} \cup \{\overline{x}_i | i \in I_1\} \tag{4.7}$$

Ainsi, l'instanciation partielle $(1, *, *, 1, *, 0)$ sera notée indifféremment $\overline{x}_1 \vee \overline{x}_4 \vee x_6$ ou $\{\overline{x}_1, \overline{x}_4, x_6\}$ ou encore parfois $\overline{x}_1\overline{x}_4x_6$. Cette dernière implique que dans la suite de la recherche les solutions explorées, et potentiellement à même de mener à une solution améliorant la valeur de fonction objectif, sont telles que $x_1 = 0$ ou $x_4 = 0$ ou $x_6 = 1$.

La clause vide $(*, \ldots, *)$, notée $\emptyset$, correspond à l'espace de recherche complet.

Deux clauses $u$ et $v$ sont dites en "clash" s'il existe exactement un littéral $x$ tel que $x \in u, \overline{x} \in v$. On appelle *résolvante* de $u$ et $v$ la clause $C$ telle que :

$$C = u \backslash \{x\} \cup v \backslash \{\overline{x}\} \tag{4.8}$$

et on note $C = u \bigtriangledown v$.

### 4.1.2 Principe

Resolution Search est basée sur le principe de *résolution par réfutation* qui consiste à démontrer qu'une valeur $\overline{z}$ est optimale pour un problème donné en prouvant qu'il n'existe pas de solution $u \in \{0,1\}^n$ telle que $oracle(u) < \overline{z}$. Cette preuve consiste en une séquence $C^1, C^2, \ldots, C^M$ de clauses telles que :

$$\forall k = 1, \ldots, M \quad oracle(C^k) \geq \overline{z} \text{ ou } C^k = C^i \bigtriangledown C^j, i, j < k \tag{4.9}$$

$$C^M = \emptyset \tag{4.10}$$

La première assertion assure que $oracle(u) \geq oracle(C^k) \geq \overline{z} \ \forall \ u \mid C^k \sqsubseteq u$ car *oracle* est non-décroissante. À fortiori, $oracle(u) \geq oracle(C^M) \geq \overline{z} \ \forall \ u \mid C^M \sqsubseteq u$. D'où $oracle(u) \geq \overline{z} \ \forall \ u$.

Resolution Search propose de générer une telle suite de clauses. À cet effet, la méthode procède de façon similaire aux backtracking intelligents, en cela que les clauses ajoutées à la séquence sont des nogoods, c.-à-d. qu'elles ne coupent pas uniquement l'instanciation partielle courante mais une sous-instanciation de celle-ci. Cependant, au contraire de ces derniers, Resolution Search prend en considération le critère d'optimisation dans le processus d'identification de la sous-instanciation en cause de l'échec ce qui ce dernier moins trivial. Chvátal propose ainsi de considérer la désinstanciation de chacune des variables de $u$ (à l'exception de la dernière) dans l'ordre inverse de leur instanciation tant que l'évaluation donnée par *oracle* satisfait à la condition requise. Au cours de cette phase de remontée (*waning phase*), la clause $u$ est ainsi itérativement remplacée par $u \backslash \{l\}$ jusqu'à ce que $oracle(u \backslash \{l\}) < \overline{z}$. Au terme de la remontée, le nogood $u$ est ajouté à la séquence de clauses et le processus de descente (*waxing phase*) est réamorcé.

Resolution Search s'attache à mémoriser une sous-famille $F$ de telles clauses dont la propriété essentielle permet d'assurer la complétude de l'algorithme tout en permettant de déterminer rapidement comment poursuivre la recherche, suite à la découverte d'un nogood. Nous décrivons ces règles en détail dans la section suivante.

### 4.1.3 Gestion de la famille des nogoods

Dans cette section, nous commençons par présenter la structure particulière de la famille $F$, puis détaillons les procédés de gestion de cette dernière, à savoir le mécanisme de poursuite de la recherche ainsi que le processus de mise à jour lorsqu'un échec est rencontré.

Soit la famille $F = \{C^1, C^2, \ldots, C^M\}$ de clauses ordonnées associées respectivement aux littéraux $l^1, l^2, \ldots, l^M$. $F$ est dite *path-like* si elle vérifie les conditions suivantes :

- le littéral associé à une clause n'apparaît que dans celle-ci : $l^i \in C^j$ si et seulement si $i = j$ ;
- la négation du littéral associé à une clause n'apparaît pas dans les clauses précédentes : si $\overline{l}^i \in C^j$ alors $j > i$ ;
- pour toute variable $x$ ne correspondant pas à un littéral associé à une clause, on ne peut avoir $x = 0$ dans une clause et $x = 1$ dans une autre : $\forall l, l \in C^i, \overline{l} \in C^j \ : \ l = l^i$ ou $l = l^j$.

À cette famille $F$, on associe une clause $u_F$ définie par :

$$u_F = (\bigcup_{k=1}^{M}(C_k \cup \overline{l}_k \backslash \{l_k\})) \tag{4.11}$$

$u_F$ consiste le point de départ de la phase de descente. À l'aide des propriétés sus-citées, il est aisé de vérifier que celle-ci est bien définie, c.-à-d. qu'il n'existe aucun littéral $l$ tel que $l \in u_F$ et $\overline{l} \in u_F$. En outre, de façon triviale, aucune clause $C^k$ de $F$ ne constitue une sous-instanciation de $u_F$ ou d'une quelconque extension de celle-ci (car $\overline{l}^k \in u_F$ et $l^k \in C^k$).

À tout instant, la famille $F$ doit conserver la propriété de path-like. Soit $\overline{z}$ la valeur de la meilleure borne supérieure connue courante (qui est mise à jour dès que l'on rencontre une solution réalisable $x$ avec $oracle(x) < \overline{z}$), $F$ n'est constituée que de clauses $C^k$, telles que l'évaluation de toute solution $x$ étendant $C^k$ vérifie $oracle(x) \geq \overline{z}$. Pour cela, les opérations suivantes sont autorisées sur $F$, à l'exclusion de toute autre :

- une clause $u$ peut être ajoutée à $F$ si $oracle(u) \geq \overline{z}$ ;
- si l'on peut effectuer la résolvante de deux clauses $C^k$ et $C^{k'}$ de $F$, alors on peut ajouter $C^k \bigtriangledown C^{k'}$ à $F$ ;
- une clause $C^k$ peut être supprimée de $F$.

De façon plus précise, si l'on considère la clause $u$ obtenue au terme d'une phase de remontée de Resolution Search, qui vérifie $oracle(u) \geq \overline{z}$ et telle que $u \sqsubseteq u'$, avec $u'$ la clause obtenue au terme d'une phase de descente et telle que $u_F \sqsubseteq u'$, alors la propriété suivante est vérifiée :

$$l \in u_F \Rightarrow \overline{l} \notin u \tag{4.12}$$

En effet, comme $u'$ constitue une extension à la fois de $u_F$ et de $u$, toute variable instanciée dans $u'$ se retrouve, dans $u_F$ ou $u$, soit instanciée à la même valeur, soit non instanciée. Cette observation nous permet de maintenir la structure particulière path-like de $F$, laquelle est mise à jour à partir de $u$ de deux façons distinctes selon que $u$ soit, ou non, une sous-instanciation de $u_F$.

#### Avec phase de descente

Ce premier cas de figure apparaît lorsque $oracle(u_F) < \overline{z}$. En conséquence, une phase de descente est amorcée au cours de laquelle au moins un littéral est ajouté à $u_F$ et, au terme de la phase de remontée consécutive, $u$ ne constitue donc pas une sous-instanciation de $u_F$. L'opération de maintien de $F$ consiste ici uniquement à rajouter cette nouvelle clause à la famille en lui associant un littéral $l \in u \backslash u_F$ :

$$M = M + 1, C^M = u, l^M = l \tag{4.13}$$

De façon triviale, cette mise-à-jour conserve le caractère path-like de $F$.

**Sans phase de descente**

Ce deuxième cas de figure est rencontré lorsque l'évaluation de $u_F$ est telle que $oracle(u_F) \geq \overline{z}$. La phase de descente n'a ici pas lieu et aucun littéral n'est ajouté à $u_F$. On obtient donc au terme de la remontée $u \sqsubseteq u_F$. $F$ est alors maintenue de la façon décrite ci-dessous.

On considère une clause $C^k$ de $F$, avec $l^k$ comme littéral associé, telle que $\overline{l}^k$ appartienne à la clause $u$. Les autres variables peuvent être soit instanciées à la même valeur dans $u$ et $C^k$, soit non instanciées dans l'une ou l'autre des deux clauses. Il est alors possible de réduire $u$ par résolvante avec $C^k$. Soient $C = u \bigtriangledown C^k$ et $C' = \bigcup_{i \neq k} \{\overline{l}^i\} \cap u$ (c.-à-d. l'ensemble des littéraux $\overline{l}^i \neq \overline{l}^k$ n'ayant pas été supprimés lors de la phase de remontée, on a :

$$C \sqsubseteq \bigcup_{i=1}^{M} (C^i \backslash \{l^i\}) \cap C' \tag{4.14}$$

De plus, toute extension de $C$ ne peut conduire à une solution améliorant la valeur de $\overline{z}$.

Pour maintenir la structure path-like de $F$, on procède de façon itérative à rebours sur les clauses de la famille. On commence par poser $C = u$ puis on effectue des résolvantes successives sur $C$ avec les clauses $C^k$ telles que $\overline{l}^k \in C$. Au final, $C$ ne contient plus aucun littéral associé à une clause quelconque de $F$ et tout littéral de $C$ apparaît dans au moins une clause $C^k$. On a donc :

$$C \sqsubseteq \bigcup_{i=1}^{M} (C^i \backslash \{l^i\}) \tag{4.15}$$

Au terme de cette série de réductions, si l'on obtient $C = \emptyset$, la procédure de Resolution Search s'achève car l'on a alors prouvé que $\overline{z}$ est la valeur optimale du problème. Dans le cas contraire, on recherche l'indice minimum $k$ tel que $C$ est inclue dans l'union des $k$ premières clauses de $F$ et un littéral $l$ apparaissant dans $C^k$ mais dans aucune autre clause précédente :

$$C \sqsubseteq \bigcup_{i=1}^{k} (C^i \backslash \{l^i\}), l \in C, l \notin C^i, \forall i \in \{1, \ldots, k-1\} \tag{4.16}$$

On remplace alors la clause $C^k$ par $C$ en lui associant le littéral $l^k = l$. On peut aisément vérifier que le caractère path-like de $F$ est maintenu suite à cette opération. En effet, d'après 4.15, $C$ ne contient aucun littéral $l^i$ ou $\overline{l}^i$ ; de plus, $\overline{l}$ et $l$ n'apparaissent dans aucune clause précédente $C^i, i < k$.

Finalement, afin de respecter la condition 1, il est nécessaire de supprimer toutes les clauses $C^i, i > k$ qui contiennent $l$.

En recalculant $u_F$, il est possible de recommencer le processus de mise à jour de la famille $F$ lorsque $l \notin u$. En effet, à l'exception de $C^k$, toute clause de la nouvelle famille $F$ appartient à l'ancienne. Tout littéral de $u_F \backslash \{\overline{l}_k\}$ appartient donc à la clause $u_F$ précédente, que nous notons $u'$. $u_F$ constitue ainsi une sous-instanciation de $u'$ de même que $u$ en est une. Les conditions sont ainsi réunies pour poursuivre la mise à jour de $F$.

**Algorithme**

L'algorithme présenté en Figure 4.1 décrit de façon formelle le processus de mise à jour de la famille path-like $F$ à la suite de la découverte d'une nouvelle clause entrante $u$ telle que $u \sqsubseteq u'$ avec $u_F \sqsubseteq u'$ et $oracle(u) \geq \overline{z}$.

---

**Début**
1. **Si** $u \sqsubseteq u_F$ **alors**
2. continuer=vrai ;
3. **Tant que** continuer **faire**
4. $C = u$
5. **Pour** $k = M, \ldots, 1$ **faire**
6. **Si** $\overline{l}^k \in C$ **alors**
7. $C = C \nabla C^k$
8. **Si** $C = \emptyset$ **alors**
9. STOP
10. $M' = \min\{k | C \subseteq C^1 \cup \ldots \cup C^k\}$
11. Soit $l \in C \backslash C^1 \cup \ldots \cup C^{M'-1}$
12. $C^{M'} = C, l^{M'} = l$ ;
13. **Pour** $k = M' + 1, \ldots, M$ **faire**
14. **Si** $l \notin C^k$ **Alors**
15. $M' = M' + 1, C^{M'} = C^k, l^{M'} = l^k$
16. $M = M'$
17. **Si** $l \in u$ **alors**
18. continuer=faux ;
19. **Sinon**
20. Soit $l \in u \backslash u_F$
21. $M = M + 1, C^M = u, l^M = l$ ;

**Fin**

---

FIG. 4.1 – *Mise à jour de la famille F*

**Exemple**

Pour illustrer le processus de gestion de la famille $F$, nous présentons ci-dessous un exemple proposé dans [Chvátal 1997].

Soit $F$ la famille path-like donnée par les clauses et littéraux associés suivants : $C^1 = x_3, l^1 = x3, C^2 = \overline{x}_2\overline{x}_4\overline{x}_6, l^2 = \overline{x}4, C^3 = \overline{x}_2\overline{x}_5\overline{x}_6, l^3 = \overline{x}_5, C^4 = \overline{x}_2\overline{x}_6x_7, l^4 = x_7, C^5 = \overline{x}_1\overline{x}_2\overline{x}_3, l^5 = \overline{x}^1, C^6 = x_1x_5x_8, l^6 = x_8$.

Soit $u = \overline{x}_3\overline{x}_6\overline{x}_8$, la clause retenue pour mettre à jour $F$.

On remarque de $u$ est une sous-instanciation de $u_F = x_1\overline{x}_2\overline{x}_3x_4x_5\overline{x}_6\overline{x}_7\overline{x}_8$. Il s'agit alors de calculer la clause $C$ à ajouter à $F$ en effectuant des résolvantes successives :

$$C = u \bigtriangledown C^6 = x_1\overline{x}_3x_5\overline{x}_6, \tag{4.17}$$
$$C = C \bigtriangledown C^5 = \overline{x}_2\overline{x}_3x_5\overline{x}_6, \tag{4.18}$$
$$C = C \bigtriangledown C^3 = \overline{x}_2\overline{x}_3\overline{x}_6, \tag{4.19}$$
$$C = C \bigtriangledown C^1 = \overline{x}_2\overline{x}_6. \tag{4.20}$$

On peut constater que $C \subseteq C^1 \cup C^2$. On remplace donc $C^2$ par $C$ et on détermine un littéral associé $l^2 \notin C^1$, par exemple $\overline{x}_2$. Afin de conserver la propriété path-like de $F$, les clauses $C^k, k > 2$ contenant le littéral $l^2$ sont supprimées et l'on obtient ainsi une famille $F$ ne contenant plus que trois clauses : $C^1 = x_3, C^2 = \overline{x}_2\overline{x}_6, C^3 = x_1x_5x_8$, associées respectivement aux littéraux $l^1 = x_3, l^2 = \overline{x}_2, l_3 = x_8$.

Une remarque concerne ici le choix du littéral. En effet, bien que $\overline{x}_6$ convenait également comme littéral associé à $C^2$, le choix de $\overline{x}_2$ s'avère ici plus judicieux car ce dernier n'appartient pas à $u$. Il est ainsi possible de poursuivre la mise-à-jour de $F$. De même que précédemment, $u \sqsubseteq u_F = x_1x_2\overline{x}_3x_5\overline{x}_6\overline{x}_8$ et l'on peut à nouveau réduire $u$ par résolvantes successives :

$$C = u \bigtriangledown C^3 = x_1\overline{x}_3x_5\overline{x}_6, \tag{4.21}$$
$$C = C \bigtriangledown C^1 = x_1x_5\overline{x}_6. \tag{4.22}$$

On remplace alors $C^3$ par $C$ et lui associe un littéral $l^3 = x_1$ par exemple ($x_5$ convient également). La famille $F$ est à présent composée des clauses $C^1 = x_3, C^2 = \overline{x}_2\overline{x}_6, C^3 = x_1x_5\overline{x}_6$ auxquelles sont associés les littéraux $l^1 = x_3, l^2 = \overline{x}_2, l^3 = x_1$.

Une fois encore, comme $x_1 \notin u$, on peut mettre à jour $F$. Cette fois, $u_F$ n'est pas une extension de $u$ et l'on ajoute donc $u$ comme quatrième clause de $F$ en y associant le littéral $\overline{x}_8 \in u \setminus u_F$. Au terme de l'algorithme, la famille $F$ est donc égale à :

$$C^1 = x_3, l^1 = x_3, \tag{4.23}$$
$$C^2 = \overline{x}_2\overline{x}_6, l^2 = \overline{x}_2, \tag{4.24}$$
$$C^3 = x_1x_5\overline{x}_6, l^3 = x_1, \tag{4.25}$$
$$C^4 = \overline{x}_3\overline{x}_6\overline{x}_8, l^3 = \overline{x}_8. \tag{4.26}$$

## 4.2 Application de Resolution Search au problème des Queens_$n^2$

### 4.2.1 Le problème des Queens_$n^2$

À partir d'un plateau d'échecs de dimension $n \times n$, nous pouvons définir un graphe $G = (V, E)$, appelé *graphe des reines*, possédant $n^2$ sommets, correspondant chacun à une case du plateau, et

dont les arcs représentent les règles de déplacement des reines : deux sommets sont connectés par un arc si les cases qu'ils représentent se situent sur la même ligne, colonne ou diagonale.

Le problème de *coloration* consiste à affecter une couleur à chaque sommet du graphe de telle façon que deux sommets adjacents possèdent des couleurs distinctes. L'objectif étudié dans le cadre de ce chapitre concerne la détermination du *nombre chromatique* d'un graphe des reines possédant $n^2$ sommets, et plus précisément, la question de savoir s'il est égal à $n$. Sachant qu'un tel graphe possède des cliques maximales de taille $n$, il s'agit alors de trouver une *$n$-coloration valide* de ce graphe. De façon plus formelle, ce problème consiste à déterminer un $n^2$-uplet $C = \{c_1, \ldots, c_{n^2}\}$ des couleurs affectées aux différents sommets sous les contraintes suivantes :

$$1 \leq c_i \leq n \quad \forall i \in V \tag{4.27}$$

$$c_i \neq c_j \quad \forall (i,j) \in E \tag{4.28}$$

Nous donnons en Figure 4.2 un certificat pour $n = 5$.

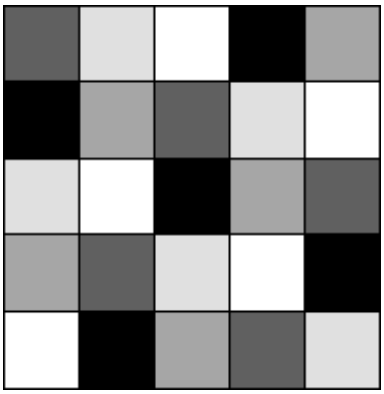

FIG. 4.2 – *Une 5-coloration valide pour Queens_*$5^2$

Un fait marquant est que peu de références concernent ce problème de coloration particulier malgré la recherche importante dévolue à la résolution de problèmes de coloration en général (voir par exemple [Hamiez 2002] pour une étude complète du sujet). Au rayon des méthodes exhaustives, nous pouvons citer l'approche de Mehrotra et Trick [Mehrotra 1996], basée sur la génération de colonnes, celle de Caramia et Dell'Olmo [Caramia 2001], qui s'appuie sur une méthodologie consistant à étendre la coloration d'une clique maximale, et, plus récemment, la procédure de backtracking développée par Vasquez [Vasquez 2004a]. La méthodologie que nous avons choisi d'employer pour résoudre ce problème consiste en une procédure combinant Resolution Search et des procédés de propagation de contraintes simples.

L'approche de Resolution Search étant désignée pour des variables binaires, nous utilisons les variables additionnelles $x_{ik}$, telles que $x_{ik} = 1$ si le sommet $i$ est coloré avec la couleur $k$, 0 sinon. Celles-ci vont nous servir d'interface entre les opérations de parcours de l'espace de recherche et

celles de gestion de la famille path-like : $C = \bigcup_{i \in I_0}(c_i \neq v_i) \cup \bigcup_{i \in I_1}(c_i = v_i) \Longrightarrow u = \bigvee_{i \in I_0} x_{iv_i} \vee \bigvee_{i \in I_1} \overline{x}_{iv_i}$.

Nous débutons par la présentation notre adaptation de Resolution Search au problème des Queens_$n^2$ en section 4.2.2, puis poursuivons par la description des points particuliers de la méthode en section 4.2.3. Nous terminons finalement par la présentation de résultats expérimentaux obtenus sur diverses instances en section 4.2.4.

### 4.2.2 La procédure hybride Resolution Search-PPC pour la résolution du problème des Queens_$n^2$

Dans cette section, nous définissons précisément l'intégration de Resolution Search dans une approche PPC pure pour la résolution du problème des Queens_$n^2$. L'approche possède la particularité de s'attacher à résoudre un problème de décision, en utilisant des algorithmes de consistance pour couper l'espace de recherche, ce qui sort du cadre strict proposé par Resolution Search. On peut cependant se ramener au problème décisionnel en posant $\overline{z} = 1$ et en attribuant la valeur 1 à *oracle* si l'instanciation partielle courante est consistante avec les contraintes du problème et 0 sinon.

Un avantage de cette démarche est qu'elle permet une identification automatique de la cause d'un échec. Resolution Search procède en effet de la même façon que les backtracking intelligents mais prend également en compte le critère d'optimisation. De fait, identifier la sous-instanciation des variables véritablement à l'origine de l'échec peut se révéler mal-aisé. Au contraire, la PPC, de par le caractère local de maintien de la consistance qu'elle sous-tend, permet de rendre cette identification immédiate. En effet, un échec se traduit par la violation d'une contrainte par l'instanciation courante. De façon triviale, il suffit alors de déterminer les variables intervenant dans ladite contrainte pour identifier un nogood. Ce processus permet de remplacer avantageusement la phase de remontée, ce qui permet une accélération notable.

L'algorithme général de la méthode est donné en Figure 4.3. Ce dernier débute par la fixation d'une des diagonales principales puis amorce un processus arborescent. À chaque nœud de l'arbre, un sommet non encore instancié est sélectionné puis affecté à une couleur non interdite de son domaine et la nouvelle affectation est propagée. La propagation consiste à déterminer la consistance de l'ensemble $C_F$ des contraintes d'affectation des variables avec les contraintes du problème initial par réduction du domaine des variables non encore instanciées. Elle est réalisée par l'appel à la fonction Propage($C_F$) qui renvoie FAUX lorsque l'ensemble $C_F$ est inconsistant avec les contraintes du problème et VRAI sinon. En cas de détection d'inconsistance, la sous-instanciation des variables responsable de l'échec est déterminée puis convertie en une clause $u$ qui est ajoutée à la famille $F$, laquelle est alors mise à jour ainsi que décrit en section 4.1.3. La clause $u_F$, et l'ensemble $C_F$ associé, sont alors déterminés et la recherche est reprise à partir de ce nouveau point. Le processus s'achève dès lors que $F$ devient vide (le problème ne possède pas de solution) ou qu'une instanciation complète est rencontrée.

**Début**
1. Fixer une diagonale principale
2. Initialiser la famille path-like : $F = \emptyset$ ;
3. $C_F = \emptyset$ ;
4. **Tant que** drapeau=Propage($C_F$)=VRAI et $\exists i \mid c_i = *$
5. Choisir un sommet $i$ tel que $c_i = *$ et sa couleur d'affectation $k \in D_i$
6. $C_F = C_F \cup (c_i = k)$
7. **Si** drapeau=VRAI **Alors**
8. Retourner $C_F$
9. **Sinon**
10. Déterminer le nogood $C \subseteq C_F$ responsable de l'échec et déterminer la clause $u$ associée
11. Mettre à jour $F$ avec $u$
12. **Si** $F = \emptyset$ **Alors**
13. Retourner $\emptyset$
14. **Sinon**
15. Déterminer $u_F$ et $C_F$
16. Aller en 4.

**Fin**

FIG. 4.3 – *Algorithme général de résolution*

### 4.2.3 Points particuliers de la méthode

Nous détaillons dans cette section les procédés mis en œuvre dans l'algorithme général, à savoir les stratégies de (dé)branchement ainsi que les techniques de propagation de contraintes utilisées.

#### Stratégies de (dé)branchement

La stratégie globale consiste à affecter les couleurs les unes après les autres aux différents sommets. Ainsi, nous tentons dans un premier temps de placer les $n$ reines de la première couleur sur l'échiquier, puis les $n$ reines de la couleur suivante, etc. Concrètement, à un nœud donné, nous sélectionnons un sommet dont le domaine contient la couleur concernée et l'y affectons. En cas de conflit entre plusieurs sommets candidats, le premier sommet dans l'ordre lexicographique est sélectionné.

En dehors de cette stratégie de branchement explicite, un autre type de branchement est impliqué par Resolution Search. Ce mécanisme de branchement implicite intervient lorsqu'il s'agit de sélectionner le littéral associé à une clause entrante de $F$ car le parcours de l'arbre de recherche se poursuit le long de la décision inverse. Lorsqu'au moins un littéral à été ajouté à $u_F$ (phase de descente), on mémorise le dernier ajouté, cause de l'échec et apparaissant donc obligatoirement dans le nogood $u$, car il satisfait trivialement à la condition $l \in u \setminus u_F$. Dans le cas inverse, le premier littéral, dans l'ordre lexicographique du terme, de $C$ satisfaisant aux conditions requises est choisi.

**Propagation de contraintes**

L'algorithme de consistance est activé lorsqu'un sommet $i$ est affecté à une couleur $k$ donnée. Il travaille sur les cliques du graphe des reines restreintes aux lignes, colonnes et diagonales.

Sachant que $i$ apparaît dans les cliques $\mathcal{C}_{i_j}, j = 1, \ldots, 4$, la règle (4.29), donnée ci-dessous, est appliquée. Celle-ci consiste à supprimer la valeur $k$ du domaine des variables $i'$ appartenant aux même cliques que $i$.

$$\forall j \quad \forall i' \in \mathcal{C}_{i_j}, i' \neq i \quad D_{i'} = D_{i'} - \{k\} \tag{4.29}$$

D'autres règles se déclenchent lorsqu'une valeur est supprimée du domaine d'un sommet. Ainsi (4.30) sert à détecter une inconsistance lorsqu'il ne reste plus suffisamment de couleurs pour colorier une clique donnée tandis que (4.31) active l'affectation du sommet $i'$ à la couleur $k'$. En effet, toute autre affectation pour $i'$ serait source d'inconsistance (il ne resterait plus assez de couleurs pour colorier la clique considérée).

$$\forall j \qquad |\bigcup_{i \in \mathcal{C}_{i_j}} D_i| < |C_{i_j}| \qquad oracle = 0 \tag{4.30}$$

$$\exists i' \in \mathcal{C}_{i_j}, k' \in D_{i'}, |(\bigcup_{i'' \neq i' \in \mathcal{C}_{i_j}} D_{i''}) \cup (D_{i'} - \{k'\})| < |C_{i_j}| \quad D_{i'} = \{k'\} \tag{4.31}$$

D'autres procédés classiques de propagation interviennent également. Notamment, lorsque le domaine d'une variable est réduit à un singleton, celle-ci est affectée à la valeur correspondante. De même, une inconsistance est détectée dès lors que le domaine d'une variable devient vide. *oracle* est alors assigné à la valeur 0.

L'algorithme retourne au terme de son exécution la valeur $VRAI$ si $oracle = 1$ et la valeur $FAUX$ sinon.

### 4.2.4 Résultats expérimentaux

Dans cette section, nous présentons les résultats obtenus sur un certain nombre d'instances. Le code a été réalisé en C++ et les expérimentations ont été réalisées sur un PC dont le processeur est cadencé à 2.4 GHz et disposant de 1 Mo de RAM.

Notre optique étant dans un premier temps de valider l'approche de Resolution Search, nous avons ainsi comparé les résultats obtenus par l'algorithme présenté précédemment avec ceux obtenus par une version de l'algorithme n'intégrant pas Resolution Search et qui consiste donc en une procédure de backtracking classique. Le schéma de branchement ainsi que les propagations de contraintes invoquées sont identiques pour les deux approches. De la même façon, le processus est stoppé dans les deux cas au bout d'une heure d'exécution.

Nous avons reporté dans le tableau 4.1 le nombre de nœuds ainsi que les temps d'exécution en secondes (entre parenthèses) obtenus par les deux approches sur chacune des instances pour $n$ variant de 5 à 11. Les instances telles que $n \leq 5$ sont résolues par propagation exclusive à la racine, celles pour lesquelles $n \geq 12$ ne sont résolues par aucune des méthodes. La dernière colonne

indique si une solution existe pour l'instance considérée.

| n | PPC+RS | Back | Solution |
|---|---|---|---|
| 5 | 1 (0) | 1 (0) | oui |
| 6 | 10 (0) | 4 (0) | non |
| 7 | 17 (0) | 24 (0) | oui |
| 8 | 631 (0) | 15333 (1) | non |
| 9 | 43669 (14) | 14482595 (576) | non |
| 10 | 2092698 (1158) | - | non |
| 11 | 1371538 (1115) | - | oui |

TAB. 4.1 – Résultats obtenus sur les instances avec $n \leq 11$

Ces premiers résultats se montrent très largement à l'avantage de Resolution Search, dès lors que le problème augmente en taille : les temps d'exécution ainsi que le nombre de nœuds sont bien moindres que ceux obtenus par la procédure de backtracking classique équivalente. Cette dernière se révèle de plus incapable de trouver de solution, ou prouver qu'il n'en existe pas, pour $n = 10$ et $n = 11$. Ces résultats sont en adéquation avec ceux obtenus par Demassey [Demassey 2003] sur le RCPSP.

Afin d'établir une comparaison plus pertinente, nous avons également comparé les performances de notre procédure avec celles obtenues par des méthodes classiques de la littérature. Nous nous confrontons ainsi aux méthodes citées en section 4.2.1.

Nous avons reporté en table 4.2, les résultats comparatifs de chacune des approches sus-citées. Nous avons ainsi indiqué les temps d'exécution (en secondes lorsqu'aucune indication contraire n'est reportée) obtenus sur chaque instance pour $n$ variant de 5 à 14. Les instances telles que $n \geq 15$ ne sont résolues par aucune des approches.

| n | PPC+RS | [Mehrotra 1996] | [Caramia 2001] | [Vasquez 2004a] |
|---|---|---|---|---|
| 5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 4 | 0 | 0 |
| 8 | 0 | 19 | 0 | 0 |
| 9 | 14 | 515 | 0 | 0 |
| 10 | 1158 | - | - | 0 |
| 11 | 1115 | - | - | 1 |
| 12 | - | - | - | 6963 |
| 13 | - | - | - | 168 heures |
| 14 | - | - | - | 131 heures |

TAB. 4.2 – Résultats comparatifs pour $n$ variant de 5 à 14

Les résultats procurés par notre procédure, s'ils se situent très largement en deçà de ceux obtenus par Vasquez, se montrent toutefois encourageants vis-à-vis des autres méthodes, et nous ont conforté dans notre objectif d'intégrer au sein du processus global de Resolution Search des

aspects heuristiques. Nous présentons ainsi au cours de la section suivante une hybridation de la méthode concernant la dégradation de l'oracle à l'aide de symétries.

## 4.3 Une version incomplète de Resolution Search pour le problème des Queens_$n^2$

L'évaluation d'une instanciation partielle donnée peut être réalisée à l'aide d'un oracle dégradé. Il s'agit alors de déterminer des critères à priori, relatifs aux caractéristiques de l'instantiation partielle courante, qui vont permettre de considérer cette dernière comme étant invalide. De fait, même si elle satisfait aux conditions requises sur $\overline{z}$, elle sera écartée de la suite de la recherche et ajoutée comme nogood à la famille $F$. Ainsi, la dégradation de l'évaluation de l'oracle permet de lever la caractéristique de complétude de Resolution Search.

Nous proposons dans cette section une variante incomplète de la méthode proposée précédemment qui s'inspire de cette technique. L'approche reprend la méthodologie employée dans [Vasquez 2004b]. Cette dernière s'appuie sur la considération de symétries pour résoudre le problème des *Queens_*$n^2$. Nous allons ainsi restreindre l'espace de recherche aux seules configurations comportant des symétries en biaisant la réponse retournée par l'*oracle*. Ce dernier sera ainsi affecté à la valeur 0 (c.-à-d. qu'il détectera un échec) en cas d'impossibilité d'affecter simultanément des sommets symétriques à leurs valeurs symétriques.

Nous commençons par présenter en section 4.3.1 les diverses règles de symétries invoquées puis exposons en section 4.3.2 les modifications apportées à l'algorithme de propagation de contraintes permettant à l'oracle de prendre en compte ces symétries. Nous exposons ensuite les résultats expérimentaux obtenus sur diverses instances en section 4.3.3. Nous concluons finalement sur une série de perspectives destinées à préciser l'orientation future de notre recherche.

### 4.3.1 Les règles de symétrie [Vasquez 2004b]

Soit un plateau d'échec de dimension $n$. Les lignes (resp. colonnes) sont numérotées par ordre croissant de la gauche vers la droite (resp. du haut vers le bas). Chaque case est ainsi représentée par un couple de coordonnées $(i, j)$ (ligne,colonne).

#### Symétrie horizontale pour $n = 2p$

L'affectation à une couleur $k$ donnée d'un sommet correspondant à la case de coordonnées $(i, j)$ est réalisée conjointement à l'affectation de la couleur $k' = sym_{\mathcal{H}}(k)$ au sommet de coordonnées $(i', j') = \mathcal{H}(i, j)$ telles que :

$$\begin{cases} i' = i \\ j' = n - 1 - j \end{cases}$$

#### Symétries horizontale et verticale pour $n = 4p$

À la symétrie horizontale précédemment présentée s'ajoute une nouvelle symétrie verticale. Ainsi, les sommets correspondant aux cases $(i, j)$, $(i', j') = \mathcal{H}(i, j)$, mais aussi $(i'', j'') = \mathcal{V}(i, j)$

et $(i''', j''') = \mathcal{H}(i'', j'') = \mathcal{V}(i', j')$ sont affectés de façon simultanée aux couleurs $k, k' = sym_{\mathcal{H}}(k), k'' = sym_{\mathcal{V}}(k)$ et $k''' = sym_{\mathcal{V}}(k')$, respectivement. L'opérateur $\mathcal{V}$ est défini comme suit :

$$\begin{cases} i' = n - 1 - i \\ j' = j \end{cases}$$

**Symétrie centrale pour** $n = 2p + 1$

L'affectation à une couleur $k$ donnée d'un sommet correspondant à la case de coordonnées $(i, j)$ est réalisée conjointement à l'affectation de la couleur $k' = sym_{\mathcal{C}}(k)$ au sommet de coordonnées $(i', j') = \mathcal{C}(i, j)$ telles que :

$$\begin{cases} i' = n - 1 - i \\ j' = n - 1 - j \end{cases}$$

**Symétries par rotations pour** $n = 4p + 1$

L'opérateur de symétrie considéré ici est une rotation $\mathcal{R}$ de $\Pi/2$ autour de la case centrale du plateau. Ainsi, les sommets $(i, j)$ et $(i', j')$, mais aussi $(i'', j'') = \mathcal{R}(i', j') = \mathcal{R}^2(i, j)$ et $(i''', j''') = \mathcal{R}(i'', j'') = \mathcal{R}^3(i, j)$ sont affectés simultanément aux couleurs $k, k' = sym_{\mathcal{R}}(k), k'' = sym_{\mathcal{R}}(k')$ et $k''' = sym_{\mathcal{R}}(k'')$, respectivement, de telle façon que :

$$\begin{cases} i' = j \\ j' = n - 1 - i \end{cases}$$

### 4.3.2 Modification de l'oracle

On appelle $SYM$ l'ensemble des opérateurs de symétrie applicables à l'instance courante. L'algorithme de consistance est renforcé afin de prendre en compte les règles qu'ils définissent. Ainsi, suite à l'instanciation d'un sommet $i$ à une couleur $k$ donnée, la règle suivante est appliquée en plus de celles exposées précédemment :

$$\forall \mathcal{O} \in SYM \quad c_{\mathcal{O}(i)} = sym_{\mathcal{O}}(k) \tag{4.32}$$

### 4.3.3 Résultats expérimentaux

Nous avons testé dans un premier temps l'approche intégrant les symétries sur les instances qui étaient résolues par la précédente version. Les résultats sont reportés en table 4.3. Nous avons indiqué pour chacune des instances le nombre de nœuds ainsi que les temps d'exécution obtenus et repris entre parenthèses les résultats fournis par la méthode originelle (sans les symétries).

| n | # nœuds | CPU |
|---|---|---|
| 5 | 0 (1) | 0 (0) |
| 6 | 0 (10) | 0 (0) |
| 7 | 8 (17) | 0 (0) |
| 8 | 16 (631) | 0 (0) |
| 9 | 1204 (43669) | 0 (14) |
| 10 | 346 (2092698) | 0 (1158) |
| 11 | 2426 (1371538) | 3 (1115) |

TAB. 4.3 – Dégradation de l'oracle par symétries

Ces résultats démontrent clairement l'apport amené par l'intégration des symétries. En effet, pour chacune des instances résolues, le nombre de nœuds ainsi que les temps d'exécution sont diminués de façon drastique. En addition, quatre instances supplémentaires ont été résolues grâce à cet ajout, dont trois sont reportées en table 4.4. La dernière (Queen_13$^2$) n'est pas indiquée du fait de l'existence d'un algorithme linéaire de résolution. Il est toutefois notable de constater qu'elle n'était pas résolue par la méthode n'intégrant pas les symétries alors qu'elle l'est à présent par la nouvelle version.

La table 4.4 reporte les résultats comparatifs de notre méthode par rapport à celle proposée par Vasquez [Vasquez 2004b], et qui consiste à incorporer au processus décrit dans [Vasquez 2004a] la méthodologie des symétries. Nous indiquons pour chacune des instances telles que $n$=12, 16 et 20, les temps d'exécution dispensés par les deux approches.

| n | RS_Sym | [Vasquez 2004b] |
|---|---|---|
| 12 | 2 | 1 |
| 16 | 34 | 1 |
| 20 | 39 | 1 |

TAB. 4.4 – Comparaison avec [Vasquez 2004b] pour Queens_12$^2$, Queens_16$^2$ et Queens_20$^2$

La comparaison de notre approche avec la méthode mise au point par Vasquez [Vasquez 2004b] est éloquente : cette dernière requiert des temps d'exécution diminués sans aucune mesure par rapport à ceux obtenus par notre procédure. De surcroît, l'approche de Vasquez permet de résoudre les instances jusqu'à $n = 28$, à l'exception de $n = 26$ et $n = 27$.

Il apparaît alors impératif d'intégrer d'autres composants heuristiques au sein de Resolution Search en vue d'accélérer de façon drastique la procédure et espérer ainsi pouvoir résoudre de nouvelles instances. Nous présentons au cours de la prochaine section quelques pistes de recherche que nous souhaitons emprunter.

## 4.4 Perspectives

Cette section est à présent consacrée aux diverses formes d'intégration de composants heuristiques dans la procédure de Resolution Search. Ainsi que présenté au cours du premier chapitre, le but principal tient à la restriction de l'espace de recherche exploré. Pour ce faire, divers procédés peuvent être invoqués. Une technique, dans l'esprit de la liste de candidats restreints, préconise l'ajout simultané de clauses dans la famille $F$. Ce processus, qui présente des points communs avec la dégradation de l'évaluation de l'oracle, pousse ce concept un peu plus avant. Nous en présentons les grandes lignes en section 4.4.1. Finalement, nous détaillons en section 4.4.2 une dernière proposition qui réside dans l'intégration au sein du processus global de Resolution Search d'une méthode heuristique.

### 4.4.1 Ajout simultané de clauses

À un nœud donné de l'arbre de recherche, de nombreuses possibilités s'offrent à nous pour étendre la solution partielle courante. Nous avons vu au cours du premier chapitre, qu'une heuristique de guidage de la recherche peut être utilisée pour orienter le parcours de l'arbre vers les solutions à priori les plus prometteuses. Celle-ci permet en outre, de classer les diverses possibilités de branchement relativement les unes par rapport aux autres. C'est de ce principe que s'inspire la liste de candidats restreints. Cette méthodologie peut aisément être étendue dans le cadre de Resolution Search.

Ainsi, au cours du processus de recherche, lorsqu'une décision est à même d'être entérinée, il peut s'agir de supprimer de la suite de la recherche les branches considérées comme peu prometteuses. Ceci est réalisé de façon naturelle par l'ajout simultané à la famille $F$ des clauses correspondant à ces instanciations partielles peu prometteuses. Il est aisé de vérifier que cet ajout simultané maintient la structure path-like de la famille.

Cette technique se rapproche de celle que nous avons proposé sur le problème des Queens_$n^2$ tout en l'étendant. En effet, le processus de dégradation de l'oracle, ainsi que nous l'avons constaté, produit le déclenchement d'une phase de remontée de Resolution Search (détection d'un échec). À l'inverse, nous nous proposons à présent de générer des nogoods tout au long du processus de descente.

Cette piste d'exploration consiste ainsi à supprimer au fur et à mesure de la recherche des régions entières de l'espace censément non prometteuses en se basant sur l'utilisation d'un critère heuristique discriminant. Pour notre problème particulier des Queens_$n^2$, il va s'agir de déterminer de tels critères.

### 4.4.2 Intégration d'une méthode heuristique au sein du processus global

Une dernière proposition d'hybridation concerne l'intégration au sein du processus global d'une méthode heuristique dans ce qui pourrait être entrevu comme un Depth-bounded Resolution Search, à l'image de DDS (cf. chapitre 1).

L'approche peut se résumer de la façon suivante : un paramètre, noté $p$, détermine la profondeur au dessus de laquelle Resolution Search est appliqué de façon conventionnelle. Une fois

cette profondeur atteinte, un processus heuristique prend alors le relais pour explorer le sous-arbre situé sous l'instanciation partielle courante. Diverses méthodes peuvent être utilisées à cet effet, de l'algorithme glouton le plus simple à une metaheuristique plus élaborée. Dans ce dernier cas toutefois, il s'agira de restreindre convenablement la recherche au sous-arbre considéré. Pour une méthode de recherche locale par exemple, l'opérateur de voisinage devra prendre en compte le fait que certaines variables ne peuvent être assignée à une valeur différant de l'affectation courante.

Un fois le sous-arbre exploré de façon plus ou moins approfondie, la clause correspondant à l'instanciation courante partielle est ajoutée à la famille de nogoods et le processus est poursuivi. Le paramétrage de $p$ permet de plus d'influer directement sur le compromis qualité/temps désiré et apporte ainsi une certaine souplesse d'utilisation.

Concernant notre problème particulier, nous nous orienterons dans un premier temps, sur l'utilisation de la méthode DSATUR mise au point par Brélaz [Brélaz 1979]. Celle-ci se révèle en effet globalement performante sur les problèmes de coloration et sa simplicité de mise en œuvre ainsi que sa rapidité constituent des atouts non négligeables. Cependant, comme tout algorithme glouton, ses performances peuvent se révéler très médiocres, c'est pourquoi il va s'agir de calibrer la valeur du paramètre $p$ de façon adéquate. Suivant les résultats obtenus par cette approche, celle-ci pourra être étendue à l'emploi d'une méthode plus élaborée pour le parcours du sous-arbre.

# Conclusion

L'optimisation combinatoire nécessite l'emploi de méthodes de résolution toujours plus adaptées en vue d'améliorer le traitement de ces problèmes hautement difficiles. Notre position a consisté au cours de ce mémoire à préconiser l'utilisation conjointe de techniques de résolution complémentaires issues de méthodologies de recherche exacte et approchée. Ainsi, nous nous sommes focalisés sur la mise au point de procédures intégrant ces deux paradigmes au sein d'un processus incomplet unique. L'essence-même du travail présenté ici est de tirer parti des avantages procurés par chacune des techniques : d'un côté, le caractère rigoureux et optimal de l'approche exacte, de l'autre, une plus grande souplesse et rapidité d'exécution de l'approche heuristique.

A cet effet, nous avons dirigé notre recherche le long de deux axes principaux. Le premier d'entre eux a consisté à intégrer une méthode exacte au sein d'un processus de recherche locale de grands voisinages. La deuxième piste explorée tend au contraire à accélérer l'exécution d'un processus exhaustif par l'intégration de biais heuristiques. Ces deux philosophies présentent un aspect quelque peu antagoniste : dans le premier cas, la recherche locale vient remplacer le caractère déterministe de l'approche exacte pour l'exploration de l'espace de recherche et l'optimalité est conservée ; dans le deuxième cas, au contraire, l'introduction de composants heuristiques permet de supprimer la complétude de la méthode exacte tandis que la systématicité dans la recherche des solutions est conservée.

De façon concrète, nous avons commencé par détailler diverses applications d'une méthode de grands voisinages. Cette dernière est basée sur la génération puis la résolution à l'aide d'une méthode exacte de sous-problèmes successifs du problème global. La première application concerne un problème d'ordonnancement académique abondamment étudié, tandis que la deuxième porte sur un problème d'affectation de fréquences plus réaliste. Les résultats obtenus sur ces deux problèmes bien distincts ont permis de souligner des caractéristiques importantes de ce type d'approche.

L'enseignement majeur à retenir de cette étude est que l'efficacité de la procédure générale dépend en grande partie de l'aptitude à générer des sous-problèmes suffisamment indépendants du problème global. Cependant, si elle permet une plus grande efficacité, qualitativement parlant, cette approche, en rendant l'appréhension des sous-problèmes plus malaisée, comporte l'inconvénient de ralentir l'exécution du processus global. Afin de pallier ce désagrément, nous avons fait appel à diverses techniques. Concernant le problème d'ordonnancement, nous avons présenté un schéma de génération "en cascade" des sous-problèmes. L'idée principale consiste à décomposer le problème global en un certain nombre de sous-problèmes possédant des connections fortes entre eux, puis à les résoudre successivement, la résolution d'un sous-problème tenant compte du résultat

de l'optimisation des sous-problèmes précédents. Cette technique compense ainsi la considération de sous-problèmes de taille réduite par la mise en place d'un lien étroit entre les sous-problèmes. Elle a permis de fait une accélération notable du processus global. Dans le cadre du problème d'affectation de fréquences, des stratégies complémentaires permettant de faciliter, et par là-même d'accélérer, la résolution des sous-problèmes ont également été mises en place. Ces dernières ont consisté notamment à réaliser des réductions de l'espace de recherche en regard de critères heuristiques ou encore à utiliser une méthode incomplète pour la résolution des sous-problèmes.

L'ensemble des techniques développées sur l'un ou l'autre des problèmes se sont globalement révélées intéressantes et laissent entrevoir diverses perspectives. Elles semblent de plus particulièrement efficaces lorsqu'elles sont couplées à une décomposition du problème global. Il pourrait ainsi être pertinent de tenter de généraliser cette approche en développant des méthodes hybrides combinant techniques de décomposition et recherche de grands voisinages et en les éprouvant sur d'autres problèmes d'optimisation combinatoire.

Concernant la deuxième piste explorée lors de ce mémoire, nous avons proposé une méthode exacte incomplète basée sur l'utilisation conjointe de la procédure de Resolution Search [Chvátal 1997] et de techniques de programmation par contraintes. L'aspect de complétude est ici annihilé par l'introduction d'un critère discriminant permettant de supprimer l'examen de régions entières de l'espace de recherche. Cette approche a été éprouvée sur un problème de coloration particulier et, malgré des performances comparatives vis-à-vis des meilleures méthodes de la littérature en demi-teinte, les résultats préliminaires sont encourageants. De nombreuses améliorations peuvent en effet être envisagées.

La première de celle-ci va consister à appliquer une méthodologie inspirée des listes de candidats restreints. Il s'agira ainsi de supprimer à l'aide d'une heuristique les régions de l'espace de recherche jugées peu susceptibles de contenir des solutions de qualité. Le point crucial de l'approche réside dans la détermination de l'heuristique discriminante utilisée, qui doit être la plus adaptée possible au problème considéré au risque d'occulter des régions contenant de bonnes solutions, voire la solution optimale. De façon générale, quelle que soit la technique d'intégration considérée, ce paramètre constitue le point critique de l'approche. Si l'on considère une deuxième forme d'intégration consistant à déléguer l'exploration d'un sous-arbre à une méthode heuristique, là encore cette dernière se doit d'être choisie judicieusement. Dans ce deuxième schéma de coopération néanmoins, un paramètre de profondeur permet de déterminer l'importance relative des deux approches exacte et heuristique et l'on peut ainsi espérer être plus à même de trouver un compromis intéressant entre rapidité d'exécution et performance qualitative. La conception d'une telle méthode, modulable au gré des besoins et des problèmes, est d'ailleurs à terme un des nos objectifs majeurs de recherche.

# Bibliographie


[Aardal 2001] Aardal K. I., Hoesel S. P. M. V., Koster A. M. C. A., Mannino C. et Sassano A., Models and solution techniques for frequency assignment problems, Rapport technique, ZIB, 2001.

[Adams 1988] Adams J., Balas E. et Zawack D., The shifting bottleneck procedure for job shop scheduling, *Management Science*, 34 :391–401, 1988.

[Ahuja 2002] Ahuja R. K., Ergun O., Orlin J. B. et Punnen A. P., A survey of very large-scale neighborhood search techniques, *Discrete Applied Mathematics*, 123(1-3) :75–102, 2002.

[Alcaraz 2001] Alcaraz J. et Maroto C., A robust genetic algorithm for resource allocation in project scheduling, *Annals of Operations Research*, 102 :83–109, 2001.

[Alvarez-Valdés 1989] Alvarez-Valdés R. et Tamarit J., Heuristic algorithms for resource-constrained project scheduling : a review and an empirical analysis, dans Słowiński R. et Weglarz J., rédacteurs, *Advances in Projet Scheduling*, pages 113–134, Elesevier, Amsterdam, 1989.

[Applegate 1991] Applegate D. et Cook B., A computational study of the job shop scheduling problem, *ORSA Journal of Computing*, 3(2) :149–156, 1991.

[Artigues 2000] Artigues C. et Roubellat F., A polynomial activity insertion algorihm in a multi-resource schedule with cumulative constraints and multiple modes, *European Journal of Operational Research*, 127(2) :297–316, 2000.

[Baar 1998] Baar T., Brucker P. et Knust S., Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem, dans Voss S., Martello S., Osman I. et Roucairol C., rédacteurs, *Meta-heuristics : Advances and Trends in Local Search Paradigms for Optimization*, pages 1–18, Kluwer, 1998.

[Baptiste 2000] Baptiste P. et Pape C. L., Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems, *Constraints*, 5 :119–139, 2000.

[Baptiste 1995] Baptiste P., Pape C. L. et Nuijten W., Constraint-based optimization and approximation for job-shop scheduling, dans *AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems*, Montréal, 1995.

[Beck 2003] Beck J. C. et Refalo P., A hybrid approach to scheduling with earliness and tardiness costs, *Annals of Operations Research*, 118(1-4) :49–71, 2003.

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

[Beldiceanu 1999] Beldiceanu N., Bourreau E., Simonis H. et Rivreau D., Introducing metaheuristics in chip, dans *3rd Metaheuristics International Conference*, Angra do Reis, Brésil, 1999.

[Bent 2001] Bent R. et Hentenryck P. V., A two-stage hybrid local search for the vehicle routing problem with time windows, Rapport technique CS-01-06, Brown University, 2001.

[Blazewicz 1983] Blazewicz J., Lenstra J. K. et Rinooy Kan A. H. G., Scheduling subject to resource constraints : classification and complexity, *Discrete Applied Mathematics*, 5 :11–24, 1983.

[Boctor 1996] Boctor F., Resource-constrained project scheduling by simulated annealing, *International Journal of Production Research*, 34(8) :2335–2351, 1996.

[Borgne 1994] Borgne L., *Automatic frequency assignment for cellular networks using local search heuristics*, Thèse de doctorat, Uppsala University, 1994.

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

[Bouju 1995] Bouju A., Boyce J. F., Dimitropoulos C. H. D., Scheidt G. V., Taylor J. G., Likas A., Papageorgiou G. et Stafylopatis A., Intelligent search for the radio links frequency assignment problem, dans *International Conference for Digital Signal Processing, DSP'95*, Limassol, Cypres, 1995.

[Bouleimen 2003] Bouleimen K. et Lecocq H., A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, *European Journal of Operational Research*, 149(2) :268–281, 2003.

[Box 1978] Box F., A heuristic technique for assigning frequencies to mobile radio nets, *IEEE Transactions on Vehicular Technology*, 27 :57–74, 1978.

[Brélaz 1979] Brélaz D., New methods to color the vertices of a graph, *Communications of the ACM*, 22 :251–256, 1979.

[Brucker 1998] Brucker P., Knust S., Schoo A. et Thiele O., A branch & bound algorithm for the resource-constrained project scheduling problem, *European Journal of Operational Research*, 107 :272–288, 1998.

[Capone 1999] Capone A. et Trubian M., Channel assignment problem in cellular systems : a new model and a tabu search algorithm, *IEEE Transactions on Vehicular Technology*, 48(4) :1252–1260, 1999.

[Caramia 2001] Caramia M. et Dell'Olmo P., Iterative coloring extension of a maximum clique, *Journal of Heuristics*, 8 :83–107, 2001.

[Carlier 1982] Carlier J., The one-machine sequencing problem, *European Journal of Operational Research*, 11 :42–47, 1982.

[Caseau 1999a] Caseau Y. et Laburthe F., Effective forget-and-extend heuristics for scheduling problems, dans *CP-AI-OR'99, Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*, Ferrara, Italy, 1999.

[Caseau 1999b] Caseau Y. et Laburthe F., Heuristics for large constrained routing problems, *Journal of Heuristics*, 5 :281–303, 1999.

[Castelino 1996] Castelino D. J., Hurley S. et Stephens N. M., A tabu search algorithm for frequency assignment, *Annals of Operations Research*, 63 :301–319, 1996.

[Chvátal 1997] Chvátal V., Resolution search, *Discrete Applied Mathematics*, 73 :81–99, 1997.

[Costa 1993] Costa D., On the use of some known methods for t-colourings of graphs, *Annals of Operations Research*, 41 :343–358, 1993.

[Crawford 1994] Crawford J. et Baker A., Experimental results on the application of satisfiability algorithms to scheduling problems, dans *12th AAAI*, 1994.

[Crawford 1993] Crawford J. M., Solving satisfiability problems using a combination of systematic and local search, *Second DIMACS Challenge : cliques, coloring, and satisfiability*, 1993.

[Crompton 1994] Crompton W., Hurley S. et stephens N. M., A parallel genetic algorithm for frequency assignment problems, dans *IMACS/IEEE International Symposium on Signal Processing, Robotics and Neural Networks*, pages 81–84, Lille, France, 1994.

[Danna 2003a] Danna E. et Perron L., Structured vs. unstructured large neighborhood search : a case study on job-shop scheduling problems with earliness and tardiness costs, dans *9th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 817–821, 2003.

[Danna 2003b] Danna E., Rothberg E. et Le Pape C., Exploring relaxation induced neighborhoods to improve mip solutions, Rapport technique, ILOG, 2003.

[Della Croce 1995] Della Croce F., Generalized pairwise interchanges and machine scheduling, *European Journal of Operational Research*, 83 :310–319, 1995.

[Della Croce 2004] Della Croce F., Ghirardi M. et Tadei R., Recovering beam search : enhancing the beam search approach for combinatorial optimization problems, *Journal of Heuristics*, 10 :89–104, 2004.

[Demassey 2003] Demassey S., *Méthodes hybrides de programmation par contraintes et programmation linéaire pour le problème d'ordonnancement de projet à contraintes de ressources*, Thèse de doctorat, Université d'Avignon et des Pays de Vaucluse, 2003.

[Dunkin 1998] Dunkin N. W., Bater J. E., Jeavons P. G. et Cohen D. A., Towards high order constraint representations for the frequency assignment problem, Rapport technique, University of London, Egham, Surrey, UK, 1998.

[Dupont 2004] Dupont A., Alvernhe E. et Vasquez M., Efficient filtering and tabu search on a consistent neighborhood for the frequency assignment problem with polarisation, *Annals of Operations Research*, 130 :179–198, 2004.

[Feo 1995] Feo T. et Resende M., Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6 :109–133, 1995.

[Fischetti 2002] Fischetti M. et Lodi A., Local branching, *Compléter*, compléter :compléter, 2002.

[Garaix 2005] Garaix T., Artigues C. et Demassey S., Bornes inférieures et supérieures pour le rcpsp, dans *6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF'05*, Tours, France, 2005.

[Gendreau 2002] Gendreau M., Pesant G. et Rousseau L.-M., Using constraint-based operators with variable neighborhood search to solve the vehicle routing problem with time windows, *Journal of Heuristics*, 8(1) :43–58, 2002.

[Ginsberg 1993] Ginsberg M. L., Dynamic backtracking, *Journal of Artificial Intelligence Research*, 1 :25–46, 1993.

[Ginsberg 1994] Ginsberg M. L. et McAllester D. A., Gsat and dynamic backtracking, dans *fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 226–237, 1994.

[Glover 1997] Glover F. et Laguna M., *Tabu search*, Kluwer Academic Publishers, Boston, 1997.

[Gonçalves 2003] Gonçalves J. et Mendes J., A random key based genetic algorithm for the resource-constrained project scheduling problem, Rapport technique, Universidade do Porto, 2003.

[Hamiez 2002] Hamiez J. P., *Coloration de graphes et planification de rencontres sprotives : heuristiques, algorithmes et analyses*, Thèse de doctorat, Université d'Angers, 2002.

[Hao 1998] Hao J.-K., Dorne R. et Galinier P., Tabu search for frequency assignment in mobile radio networks, *Journal of Heuristics*, 4 :47–62, 1998.

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

[Hartmann 1998] Hartmann S., A competitive genetic algorithm for resource-constrained project scheduling, *Naval Research Logistics*, 45 :733–750, 1998.

[Hartmann 2002] Hartmann S., A self-adapting genetic algorithm for project scheduling under resource constraints, *Naval Research Logistics*, 49 :433–448, 2002.

[Hartmann 2000] Hartmann S. et Kolisch R., Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem, *European Journal of Operational Research*, 127(2) :297–316, 2000.

[Harvey 1994] Harvey W. D., Search and job shop scheduling, Rapport technique, CIRL, University of Oregon, 1994.

[Harvey 1995a] Harvey W. D., *Nonsystematic backtracking search*, Thèse de doctorat, Stanford University, 1995.

[Harvey 1995b] Harvey W. D. et Ginsberg M. L., Limited discrepancy search, dans *14th Internation Joint Conference on Artificial Intelligence*, pages 607–613, 1995.

[Hogg 1996] Hogg T., Huberman B. A. et Williams C. P., Phase transitions and the search problem, *Artificial Intelligence*, 81(1-2) :1–15, 1996.

[Jussien 1999] Jussien N. et Lhomme O., The path-repair algorithm, dans *Workshop on Large Scale Combinatorial Optimization and Constraints, Electronic Notes in Discrete Mathematics vol. 4*, 1999.

[Klein 1999] Klein R. et Scholl A., Computing lower bound by destructive improvement : An application to resource-constrained project scheduling, *European Journal of Operational Research*, 112 :322–346, 1999.

[Knälmann 1994] Knälmann A. et Quellmalz A., Solving the frequency assignment problem with simulated annealing, *IEEE conference publication*, 396 :233–240, 1994.

[Kochetov 2003a] Kochetov Y. et Stolyar A., Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem, dans *3rd International Workshop of Computer Science and Information Technologies*, Russie, 2003.

[Kochetov 2003b] Kochetov Y. et Stolyar A., Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem, dans *Proceedings of the 3rd International Workshop of Computer Science and Information Technologies*, Russia, 2003.

[Kolen 1999] Kolen A. W. J., A genetic algorithm for frequency assignment, Rapport technique, Universiteit Maastricht, 1999.

[Kolisch 1996] Kolisch R., Serial and parallel resource-constrained project scheduling methods revisited : Theory and computation, *European Journal of Operational Research*, 90 :320–333, 1996.

[Kolisch 1998] Kolisch R., A.Sprecher et Drexl A., Benchmark instances for project scheduling problems, *Handbook on recent Advances in Project Scheduling*, pages 197–212, 1998.

[Kolisch 1999] Kolisch R. et Hartmann S., Heuristic algorithms for the resource-constrained project scheduling problem : Classification and computational analysis, dans Weglarz J., rédacteur, *Project Scheduling : Recent Models, Algorithms and Applications*, pages 147–178, Kluwer, 1999.

[Kolisch 2004] Kolisch R. et Hartmann S., Experimental investigation of heuristics for resource-constrained project scheduling : an update, Rapport technique, Technical University of Munich, 2004.

[Kolisch 2001] Kolisch R. et Padman R., An integrated survey of deterministic project scheduling, *Omega*, 29(3) :249–272, 2001.

[Korf 1985] Korf R., Depth-first iterative deepening : an optimal admissible tree search, *Artificial Intelligence*, 27(1) :97–109, 1985.

[Langley 1992] Langley P., Systematic and nonsystematic search strategies, dans *Artificial Intelligence Planning Systems : Proceedings of the First International Conference*, 1992.

[Lhomme 2004] Lhomme O., Amortized random backtracking, *Annals of Operations Research*, 130 :143–161, 2004.

[Li 1992] Li K. Y. et Willis R. J., An iterative scheduling technique for resource-constrained project scheduling, *European Journal of Operational Research*, 56 :370–379, 1992.

[Manezzio 2000] Manezzio V. et Carbonaro A., An ants heuristic for the frequency assignment problem, *Future Generation Computer Systems*, 16 :927–935, 2000.

[Mannino 2003] Mannino C. et Sassano A., An enumerative algorithm for the frequency assignment problem, *Discrete Applied Mathematics*, 129(1) :155–169, 2003.

[Mausser 1997] Mausser H. et Lawrence S., Exploiting block structure to improve resource-constrained project schedules, dans Glover F., Osman I. et Kelley J., rédacteurs, *Metaheuritics 1995 : state of the art*, Kluwer, Maryland, 1997.

[Mautor 1997] Mautor T. et Michelon P., Mimausa : A new hybrid method combining exact solution and local search, dans *MIC'97, 2nd Metaheuristics International Conference*, Sophia Antipolis, France, 1997.

[Mautor 2001] Mautor T. et Michelon P., Mimausa : an application of referent domain optimization, Rapport technique, Laboratoire d'Informatique d'Avignon, 2001.

[Mehrotra 1996] Mehrotra A. et Trick M. A., A column generation approach for graph coloring, *INFORMS Journal of Computing*, 8(4) :344–354, 1996.

[Merkle 2002] Merkle D., Middendorf M. et Schmeck H., Ant colony optimization for resource-constrained project scheduling, *IEEE Transactions on Evolutionary Computation*, 6 :333–346, 2002.

[Mladenović 1997] Mladenović N. et Hansen P., Variable neighborhood search, *Computers & Operations Research*, 24 :1097–1100, 1997.

[Montemanni 2002] Montemanni R., Smith D. H. et Allen S. M., An ants algorithm for the minimum span frequency assignment problem with multiple interference, *IEEE Transactions on Vehicular Technology*, 51(5) :949–953, 2002.

[Nonobe 2002] Nonobe K. et Ibaraki T., Formulation and tabu search algorithm for the resource-constrained projet scheduling problem (rcpsp), dans Ribeiro C. C. et Hansen P., rédacteurs, *Essays and surveys in metaheuristics*, pages 147–178, Kluwer Academic Publishers, 2002.

[Oliva 2003] Oliva C., Allocation de fréquences par échantillonnage de gibbs, recuit simulé et apprentissage par renforcement, dans *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2003*, Avignon, France, 2003.

[Ow 1988] Ow P. S. et Morton T. E., Filtered beam search in scheduling, *International Journal of Production Research*, 26 :297–307, 1988.

[Ow 1989] Ow P. S. et Morton T. E., The single machine early-tardy problem, *Management Science*, 35 :177–191, 1989.

[Özdamar 1996] Özdamar L. et Ulusoy G., A note on an iterative forward/backward scheduling technique with reference to a procedure by li and willis, *European Journal of Operational Research*, 89 :400–407, 1996.

[Palpant 2002] Palpant M., Artigues C. et Michelon P., A heuristic for solving the frequency assignment problem, dans *XI congrès latino-iberico américain de recherche opérationnelle CLAIO*, Concepcion, Chili, 2002.

[Palpant 2003a] Palpant M., Artigues C. et Michelon P., A lns method for solving the resource-constrained project scheduling problem, dans *5th metaheuristic International Conference, MIC 2003*, pages 59.1–59.6, Kyoto, Japan, 2003.

[Palpant 2004] Palpant M., Artigues C. et Michelon P., Lssper : solving the resource-constrained project scheduling problem with large neighbourhood search, *Annals of Operations Research*, 31(1) :237–257, 2004.

[Palpant 2003b] Palpant M., Oliva C., Artigues C., Michelon P., Didi Biha M. et Defaix T., Affectation de fréquences avec sommation des perturbateurs : modèles et heuristiques, dans *4ème Conférence Francophone de MOdélisation et SIMulation, MOSIM'03*, pages 299–304, Toulouse, France, 2003.

[Park 1996] Park T. et Lee C. Y., Application of the graph coloring algorithm to the frequency assignment problem, *Journal of the Operations Research Society of Japan*, 39 :258–265, 1996.

[Patterson 1984] Patterson J., A comparison of exact approaches for solving the multiple constrained resource project scheduling problem, *Management Science*, 30(7) :854–867, 1984.

[Pesant 1999] Pesant G. et Gendreau M., A constraint programming framework for local search methods, *Journal of Heuristics*, 5(3) :255–279, 1999.

[Pinson 1994] Pinson E., Prins C. et Rullier F., Using tabu search for solving the resource-constrained project scheduling problem, dans *International Workshop on Project Management and Scheduling, PMS'94*, pages 102–106, Louvain, Belgique, 1994.

[Prais 1998] Prais M. et Ribeiro C., Reactive grasp : an application to a matrix decomposition problem in tdma traffic assignment, Rapport technique, Catholic University of Rio de Janeiro, Department of Computer Science, 1998.

[Prestwich 2004] Prestwich S., Incomplete dynamic backtracking for linear pseudo-boolean problems, *Annals of Operations Research*, 130 :57–73, 2004.

[Pritsker 1969] Pritsker A. A. B., Waters L. J. et Wolfe P. M., Multiproject scheduling with limited resources : a zero-one programming approach, *Management Science*, 16 :93–107, 1969.

[Rios Solis 2005] Rios Solis Y. et Sourd F., Voisinage exponentiel en ordonnancement avec pénalités d'avance-retard, dans *6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF'05*, Tours, France, 2005.

[Russell 1995] Russell R., Hybrid heuristics for the vehicle routing problem with time windows, *TRansportation Science*, 29 :156–166, 1995.

[Sarzeaud 2003] Sarzeaud O., Allocation de fréquences par échantillonnage de gibbs, recuit simulé et apprentissage par renforcement, dans *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2003*, Avignon, France, 2003.

[Schaerf 1997] Schaerf A., Combining local search and look-ahead for scheduling and constraint satisfaction problems, dans *15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 1254–1259, 1997.

[Schirmer 2000] Schirmer A., Case-based reasoning and improved adaptive search for project scheduling, *Naval Research Logistics*, 47 :201–222, 2000.

[Shaw 1998] Shaw P., Using constraint programming and local search methods to solve vehicle routing problems, dans *Principles and Practice of Constraint Programming CP-98*, tome 1520 de *Lecture Notes in Computer Science*, pages 417–431, Pisa, Italy, 1998.

[Sivarajan 1989] Sivarajan K. N., McEliece R. J. et Ketchum J. W., Channel assignment in cellular radio, dans *39th IEEE Vehicular Technology Conference*, pages 846–850, 1989.

[Slowinski 1994] Slowinski R., Soniewicki B. et Weglarz J., Dss for multiobjective project scheduling, *European Journal of Operational Research*, 79 :220–229, 1994.

[Smith 1998] Smith D. H., Hurley S. et Thiel S. U., Improving heuristics for the frequency assignment problem, *European Journal of Operational Research*, 107 :76–86, 1998.

[Smith 1993] Smith S. et Cheng C., Slack-based heuristics for constraint satisfaction scheduling, dans *11th National Conference on Artificial Intelligence*, 1993.

[Sourd 2001] Sourd F., Scheduling tasks on unrelated machines : Large neighborhood improvement procedures, *Journal of Heuristics*, 7(6) :519–531, 2001.

[Sprecher 1996] Sprecher A., Solving the rcpsp efficiently at modest memory requirements, Rapport technique, Université de Kiel, 1996.

[Sprecher 2002] Sprecher A., Network decomposition techniques for resource-constrained project scheduling, *Journal of the Operational Research Society*, 53 :405–414, 2002.

[Sung 1997] Sung C. W. et Wong W. S., Sequential packing algorithm for channel assignment under cochannel and adjacent-channel interference constraint, *IEEE transactions on Vehicular Technology*, 46(3) :676–686, 1997.

[Taillard 2002] Taillard E. et Voss S., Popmusic - partial optimization metaheuristic under special intensification conditions, dans Ribeiro C. et Hansen P., rédacteurs, *Essays and Surveys in Metaheuristics*, pages 613–629, Kluwer, Boston, 2002.

[Toklu 2002] Toklu Y. C., Application of genetic algorithms to construction scheduling with or without resource constraints, *Canadian Journal of Civil Engineering*, 29 :421–429, 2002.

[Tormos 2001] Tormos P. et Lova A., A competitive heuristic solution technique for resource-constrained project scheduling, *Annals of Operations Research*, 102 :65–81, 2001.

[Tsang 1998] Tsang E. et Voudouris C., Solving the radio link assignment problem using guided local search, dans *NATO Symposium on Radio Length Frequency Assigment*, Aalborg, Danemark, 1998.

[Vaessens 1996] Vaessens R. J. M., Aarts E. H. L. et Lenstra J. K., Job shop scheduling by local search, *INFORMS Journal on Computing*, 8 :302–317, 1996.

[Valenzuela 1998] Valenzuela C., Hurley S. et Smith D. H., A permutation based genetic algorithm for minimum span frequency assignment, *Lecture Notes in Computer Science*, 1498 :907–916, 1998.

[Valls 2000] Valls V., Ballestin F. et Quintanilla M. S., Resource-constraint project scheduling : A critical activity reordering heuristic, dans *PMS'2000, 7th International Workshop on Project Management and Scheduling*, pages 282–283, Osnabruck, Germany, 2000.

[Valls 2002] Valls V., Ballestin F. et Quintanilla M. S., A hybrid genetic algorithm for the rcpsp with the peak crossover operator, dans *Eighth International Workshop on Project Management and Scheduling, PMS 2002*, pages 368–370, Valencia, Spain, 2002.

[Valls 2004a] Valls V., Ballestin F. et Quintanilla M. S., Justification and rcpsp : a technique that pays, *European Journal of Operational Research*, 2004.

[Valls 2004b] Valls V., Ballestin F. et Quintanilla M. S., A population-based approach to the resource-constrained project scheduling problem, *Annals of Operations Research*, 131(1-4) :305–324, 2004.

[Vasquez 2004a] Vasquez M., New results on the queens_$n^2$ graph coloring problem, *Journal of Heuristics*, 10 :407–413, 2004.

[Vasquez 2004b] Vasquez M., Using symmetries for coloring queen graphs, dans *16th European Conference on Artificial Intelligence, ECAI 2004*, Valence, Espagne, 2004.

[Vasquez 2003] Vasquez M., Dupont A. et Habet D., Consistency checking within local search applied to the frequency assignment with polarization problem, *RAIRO Operations Research*, 37 :311–323, 2003.

[Verfaillie 1996] Verfaillie G., Lemaître M. et Schiex T., Russian doll search for solving constraint optimization problems, dans *13th National Conference on Artificial Intelligence (AAAI-96)*, pages 181–187, Portland, OR, USA, 1996.

[Vlasak 2003] Vlasak J. et Vasquez M., Résolution du problème d'attribution de fréquences avec sommation de perturbateurs, dans *5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2003*, Avignon, France, 2003.

[Walsh 1997] Walsh T., Depth-bounded discrepancy search, dans *15th International Joint Conference on Artificial Intelligence*, pages 1382–1387, Nagoya, Japon, 1997.

[Wang 1996] Wang W. et Rushforth C. K., An adaptive local-search algorithm for the channel-assignment problem (cap), *IEEE Transactions on Vehicular Technology*, 45 :459–466, 1996.

[Zaloom 1971] Zaloom V., On the resource-constrained project scheduling problem, *IEEE Transactions*, 3(4) :302–305, 1971.

[Zerovnik 1997] Zerovnik J., Experiments with a randomized algorithm for a frequency assignment problem, Rapport technique, Ecole Normale Supérieure de Lyon, 1997.

[Zoellner 1977] Zoellner J. A. et Beall C. L., A breakthrough in spectrum conserving frequency assignment technology, *IEEE Transactions on Electromagnetic Compatibility*, 19 :313–319, 1977.

# Resumé

Le travail présenté dans ce mémoire est focalisé sur la résolution de problèmes d'optimisation combinatoire hautement difficiles par le biais de méthodes incomplètes intégrant des paradigmes issus de deux approches bien distinctes, à savoir recherche exacte et approchée. L'intérêt d'une telle coopération est de tirer parti des avantages complémentaires que peuvent apporter l'un et l'autre des composants : optimalité et déterminisme de la composante exacte, rapidité et côté moins systématique de la composante heuristique.

À cet effet, deux méthodologies distinctes ont été abordées : d'une part l'intégration d'une procédure exaustive au sein d'une méthode de recherche de grands voisinages appelée LSSPER (Local Search (with) Sub-Problem Exact Resolution), d'une autre, l'utilisation d'un critère heuristique discrimant pour réduire la taille de l'espace de recherche exploré par une méthode complète dérivée de Resolution Search [Chvatal97]. Pour chacune de ces approches, une validation expérimentale a été réalisée sur divers problèmes académiques ou applicatifs (ordonnancement de projet sous contraintes de ressources, affectation de fréquences, coloration de graphes). Les résultats obtenus, s'ils ne sont pas toujours à la hauteur des meilleurs résultats pouvant être recensés dans la littérature, semblent à tout le moins exhiber l'intérêt de telles méthodologies et laissent entrevoir des perspectives de recherche aussi diverses que prometteuses.

*Mots clés : optimisation combinatoire, recherche exacte et approchée, schémas d'intégration.*