THÈSE

de l'Université de Franche-Comté
et de l' Université de Technologie de Belfort-Montbéliard

numéro attribué par la bibliothèque

pour obtenir le grade de

DOCTEUR

DISCIPLINE : **INFORMATIQUE**

UNE APPROCHE ORGANISATIONNELLE ET MULTI-AGENT POUR LA MODÉLISATION ET L'IMPLANTATION DE MÉTAHEURISTIQUES

Application aux problèmes d'optimisation de réseaux de transports

par

David MEIGNAN
Laboratoire Systèmes et Transport
Université de Technologie de Belfort-Montbéliard

Soutenue le 8 décembre 2008 devant le Jury composé de :

Marie-Pierre Gleizes	Rapporteur	Professeur à l'Université Paul Sabatier de Toulouse
François Charpillet	Rapporteur	Habilité à diriger des recherches à l'INRIA Nancy
Olivier Boissier	Examinateur	Professeur à l'ENSM de Saint-Etienne
Christian Prins	Examinateur	Professeur à l'Université de Technologie de Troyes
Yassine Ruichek	Examinateur	Professeur à l'UTBM
Abderrafiâa Koukam	Directeur de Thèse	Professeur à l'UTBM
Jean-Charles Créput	Co-Directeur de Thèse	Maître de conférences à l'UTBM

Remerciements

Je tiens à remercier en tout premier lieu Abderrafiâa Koukam mon directeur de thèse, ainsi que Jean-Charles Créput mon co-directeur pour l'attention qu'ils ont porté à mon travail durant ces trois années. Je leur suis profondément reconnaissant pour leur aide, leur disponibilité et leur patience.

Je remercie Marie-Pierre Gleizes et François Charpillet pour avoir accepté de rapporter sur cette thèse et pour leurs précieux conseils qui ont contribué à améliorer la qualité de mon manuscrit. Merci également à Olivier Boissier, Christian Prins et Yassine Ruichek d'avoir accepté de participer à mon jury.

J'ai particulièrement apprécié l'ambiance chaleureuse de travail qui règne au laboratoire Systèmes et Transports. Ainsi, je remercie tous les enseignants et doctorants que j'ai pu côtoyer durant ma thèse. Je remercie plus particulièrement, Olivier Simonin pour m'avoir initié au monde de la recherche, Olivier Lamotte pour sa bonne humeur inaltérable, Stéphane Galland ma référence en matière de programmation, Vincent Hilaire pour sa logique implacable et ses "juste fait le". Merci également à Jun Hu, Alexandre Gondran et Sergio Nogueira.

Je suis persuadé qu'une thèse est un travail aussi professionnel que personnel, ainsi je tiens à remercier mes amis et ma famille qui ont supporté mes longs moments de réflexion. Merci à Mikael, Renan, Davy, Nadège, Simon, Arthur, Virginie, Yiorgos et Sebastiàn.

Merci enfin à mes maîtres à penser et amis, Sana Moujahed et Nicolas Gaud.

À Marylène.

Sommaire

Sommaire des définitions

Sommaire des Figures

PREMIÈRE PARTIE

Contexte

CHAPITRE 1

INTRODUCTION

1.1 Contexte général

Il semble que dans la majorité de ses activités, l'homme tend à vouloir maximiser les bénéfices ou de manière équivalente à minimiser les pertes. Un exemple quotidien de ce fait, est la manière dont on choisit un trajet pour se déplacer d'un endroit à un autre. De manière naturelle, nous cherchons soit à minimiser la durée ou parfois la distance, ou encore, maximiser le confort lié au trajet. Ainsi, pour répondre à ce besoin constant d'optimisation, l'homme a fourni et fournira encore beaucoup d'efforts pour le développement d'outils formels aidant à atteindre l'optimalité relativement aux décisions qu'il doit prendre. L'objectif de ces outils d'optimisation est de fournir une solution de qualité à un problème donné en un temps raisonnable.

Un grand nombre de problèmes d'optimisation ayant un impact économique important sont connus pour être particulièrement difficiles à résoudre. Par conséquent, la production d'une solution optimale à ces problèmes requiert beaucoup trop de temps. Pour ce type de problème, nous avons recours aux méthodes heuristiques, qui sont des algorithmes spécifiques au problème traité permettant d'obtenir une solution de bonne qualité en un temps raisonnable. Même si ces heuristiques ne fournissent aucune garantie théoriquement prouvée sur la qualité du résultat, elles sont exploitées avec succès sur des problèmes pratiques majeurs. Le principal inconvénient des heuristiques concerne leur aspect spécifique aux problèmes. Le développement d'une heuristique est, en conséquence, une tâche difficile nécessitant une expertise importante dans le domaine du problème traité comme dans le domaine des heuristiques. L'apparition des métaheuristiques a réduit cette difficulté en proposant des schémas de résolution heuristique pouvant être adaptés à différents problèmes d'optimisation. La métaheuristique fournit un modèle générique de résolution nécessitant d'être spécialisé pour le problème particulier considéré. Lorsqu'une métaheuristique a été ainsi spécialisée, nous disons encore instanciée, celle-ci devient en quelque sorte une heuristique particulière pouvant résoudre effectivement un problème d'optimisation.

Les enjeux des métaheuristiques que nous identifions sont :

– *la performance :* Tout comme les heuristiques, l'objectif principal d'une métaheuristique est de fournir, une fois instanciée, une méthode de résolution performante en

termes de qualité de résultat et de temps de résolution. L'exécution distribuée de métaheuristiques est une solution intéressante pour améliorer les performances de ces méthodes. Cette problématique est largement encouragée avec l'apparition récente des architectures de processeur multi-core permettant l'exécution simultanée de plusieurs processus sur un même processeur d'ordinateur.

– *la robustesse :* Une fois instanciée, une métaheuristique doit être capable de résoudre aussi efficacement que possible différentes instances d'un même problème. Il s'agit non seulement de pouvoir traiter de manière efficace des instances de différentes tailles, mais aussi de traiter des instances dont la configuration est différente, tout cela en évitant à l'utilisateur d'intervenir sur les paramètres de la méthode de résolution. Différentes versions dites adaptatives ou auto-adaptatives ont été conçues pour assurer cette robustesse, principalement en modifiant dynamiquement les paramètres de la méthode.
– *la simplicité de mise en oeuvre :* L'utilisation d'une métaheuristique doit permettre de simplifier le développement par rapport à la mise en oeuvre d'une heuristique spécifique au problème traité. La facilité de compréhension est aussi un élément essentiel des métaheuristiques. Ainsi, la métaphore joue un rôle important dans les métaheuristiques et peut expliquer le succès de certaines d'entre elles.
– *la flexibilité :* Par définition une métaheuristique est un schéma générique de résolution pouvant s'appliquer à différents problèmes d'optimisation. La flexibilité se définit par la capacité d'une métaheuristique de traiter différents problèmes en conservant ses performances.

L'application d'une métaheuristique consiste généralement à sélectionner le schéma de métaheuristique proprement dit, puis l'implanter en intégrant directement les caractéristiques du problème à traiter. Peu d'outils de conception permettent d'assister ce développement, et cette démarche, qui souvent repose sur l'expérience du concepteur, ne permet pas une bonne réutilisation, est sujette aux erreurs, et est finalement coûteuse en temps de développement. Afin de remédier à ces problèmes, il nous a semblé pertinent de proposer des outils qui encouragent la modularité et la réutilisation. De plus, nous considérons qu'il peut être avantageux d'aborder la question de la distribution et de l'intégration de mécanismes d'adaptation dans les métaheuristiques. Aussi, pour répondre d'une manière générale à ces différentes questions et contribuer aux différents enjeux que nous venons d'évoquer, nous proposons dans cette thèse d'adopter le paradigme multi-agent pour développer des outils de conception et d'implantation des métaheuristiques.

Les systèmes multi-agents forment un paradigme pour la conception des systèmes complexes et proposent des outils pour les analyser, les concevoir et les implanter. L'approche agent considère les systèmes comme des sociétés composées d'entités autonomes et indépendantes, appelées agents, qui interagissent en vue de résoudre un problème ou de réaliser collectivement une tâche. Cette approche semble intéressante dans le cadre des métaheu-

ristiques, tout d'abord pour aborder la distribution et l'adaptation, mais aussi pour apporter des outils conceptuels et méthodologiques permettant de les analyser et de les modéliser. Nos travaux se situent donc à l'intersection des domaines de l'optimisation combinatoire et des systèmes multi-agents. Ils s'inscrivent dans la continuité des propositions de modèles ou de frameworks pour les métaheuristiques. Cependant, en plus de proposer un cadre de modélisation commun aux différentes métaheuristiques, l'accent est mis sur la potentialité de mise en œuvre distribuée des approches et sur l'utilisation de techniques d'apprentissage artificiel permettant de guider et d'adapter dynamiquement des méthodes de recherche.

1.2 Objectifs de nos travaux

L'objectif de cette thèse peut être résumé ainsi :

> Proposer des outils d'analyse, de conception et d'implantation des métaheuristiques pour l'optimisation combinatoire en s'appuyant sur une approche orientée-agent.

La démarche que nous adoptons pour atteindre cet objectif est la suivante. Tout d'abord, pour appliquer la démarche de conception des systèmes multi-agents aux métaheuristiques nous définissons un cadre commun de modélisation de ces dernières en proposant un framework fondé sur une approche organisationnelle. Ensuite, pour valider cet outil et transférer les concepts qui nous semblent pertinents des systèmes multi-agents aux métaheuristiques, nous proposons une métaheuristique utilisant la métaphore de la coalition. Enfin, nous mettons en oeuvre cette métaheuristique pour traiter deux problèmes d'optimisation combinatoire afin d'évaluer expérimentalement l'approche que nous proposons.

1.2.1 *AMF* : Un framework orienté-agent pour la conception de métaheuristiques

Pour intégrer l'approche multi-agent dans la conception de métaheuristiques, nous proposons un framework nommé *AMF* pour "*Agent Metaheuristic Framework*". Un framework est un ensemble d'outils facilitant le développement d'application. Dans le cadre des métaheuristiques, il doit fournir un modèle ou un squelette assez générique pour être réutilisé lors de la conception de différentes métaheuristiques. Ainsi, en donnant un cadre de modélisation commun, un framework doit faciliter l'analyse des algorithmes existants, et assister la conception de nouveaux algorithmes.

AMF se distingue des frameworks existant en introduisant les concepts permettant de traiter la distribution et l'adaptation dans les métaheuristiques. Nous exploitons pour cela plusieurs notions issues du domaine des systèmes multi-agents. Tout d'abord, le modèle de

métaheuristique que nous proposons dans le cadre d'*AMF* est un modèle organisationnel qui décrit le système sous la forme d'une organisation composée de rôles en interaction. L'intérêt de cette approche, actuellement utilisée dans les systèmes multi-agents, est de pouvoir décrire un système aussi bien comme un tout, le système multi-agent, que comme un assemblage de composants, les agents. De plus, cette approche permet de distinguer l'analyse des fonctions du système, de l'analyse de son architecture. Enfin, l'approche organisationnelle encourage la modularité et la réutilisation des modèles. Nous proposons en complément de ce modèle un guide méthodologique. Il définit un ensemble d'étapes permettant de passer du modèle organisationnel à une méthode d'optimisation exprimée en termes d'agent.

En proposant ce framework, nous souhaitons, d'une part, proposer une démarche de conception de métaheuristiques suivant un cadre commun de modélisation, et d'autre part, encourager la distribution et l'utilisation de mécanismes d'adaptation.

1.2.2 *CBM* : Une métaheuristique multi-agent fondée sur la métaphore de la coalition

Pour valider le framework *AMF* et transférer les concepts qui nous semblent pertinents des systèmes multi-agents aux métaheuristiques, nous proposons une métaheuristique fondée sur la métaphore de la coalition *CBM* (*Coalition-Based Metaheuristic*). Dans cette métaheuristique, la recherche de solution est effectuée par un ensemble d'agents regroupés dans une coalition. Chaque agent est capable d'effectuer indépendamment des autres une recherche dans l'espace des solutions à l'aide d'opérateurs de déplacement dans un voisinage de la solution courante et d'adapter sa stratégie par apprentissage par renforcement. Des mécanismes de coopération entre agents doivent permettre d'améliorer l'efficacité de la recherche. L'objectif est d'exploiter les différents aspects des systèmes multi-agents pour concevoir une méthode efficace, robuste et flexible. Ces aspects apparaissent, tout d'abord, dans la distribution et l'emploi de mécanismes de coopération entre agents, et ensuite, dans l'usage de techniques issues de l'apprentissage artificiel. En concevant *CBM* comme un système décentralisé où chaque agent possède une certaine autonomie, il devient possible de la mettre œuvre de façon parallèle. D'autre part, cette distribution permet d'envisager différents modes de coopération entre agents que sont le partage de solutions, la mise en concurrence de différentes stratégies de recherche et l'apprentissage par mimétisme.

Dans cette approche, des mécanismes d'apprentissage sont exploités afin d'adapter dynamiquement la stratégie de recherche. Cette adaptation consiste dans notre méthode à trouver l'ordonnancement d'opérateurs le plus approprié à l'instance de problème traitée. Cela est réalisé par une méthode d'apprentissage par renforcement. Ce mécanisme est combiné à une méthode collective d'apprentissage par mimétisme. L'efficacité de notre approche est évaluée expérimentalement en traitant deux problèmes d'optimisation combinatoire : un problème de tournées de véhicules et un problème de positionnement.

1.3 Plan de la thèse

Après ce bref exposé de nos propositions, cette section introduit le plan de la thèse. Le mémoire est organisé en trois parties. La première présente un état de l'art sur les métaheuristiques et les systèmes multi-agents. La seconde partie introduit le framework ainsi que la métaheuristique que nous proposons. La dernière partie présente les résultats expérimentaux de notre approche sur deux problèmes d'optimisation. Cette structuration est illustrée par la figure 1.1.

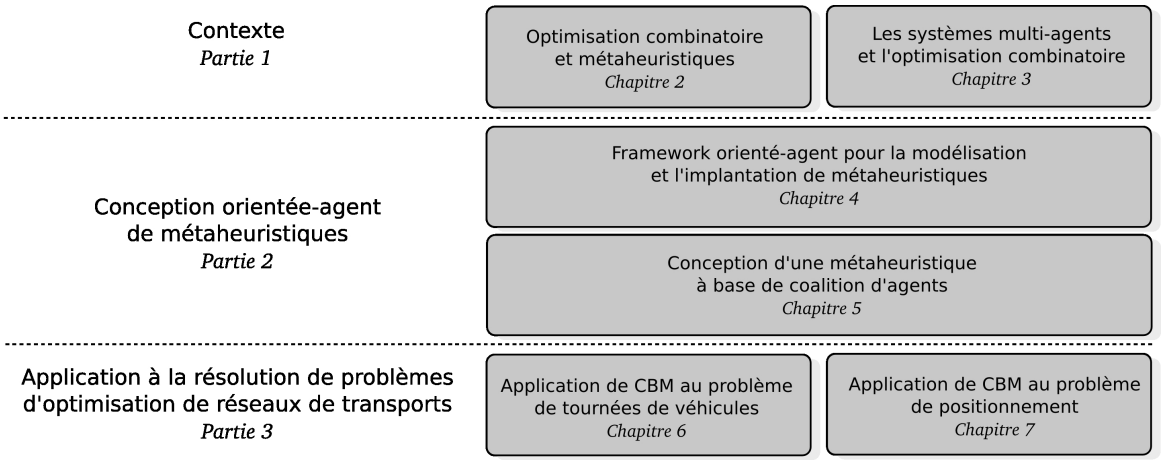

Figure 1.1: Structuration de la thèse

Chapitre 2 - Optimisation combinatoire et métaheuristiques Dans un premier temps, ce chapitre donne quelques définitions sur l'optimisation et les métaheuristiques. Ensuite, un panorama des métaheuristiques est dressé. Enfin, un état de l'art dédié aux modèles et frameworks de métaheuristiques est présenté.

Chapitre 3 - Les systèmes multi-agents et l'optimisation combinatoire Nous nous intéressons dans ce chapitre à l'utilisation des systèmes multi-agents dans le domaine de l'optimisation. Pour cela, quelques définitions sur les systèmes multi-agents sont énoncées, puis nous donnons une vue d'ensemble des aspects multi-agents présents dans les heuristiques et les métaheuristiques. Ce chapitre permet de mettre en avant l'intérêt des systèmes multi-agents dans la conception de métaheuristiques.

Chapitre 4 - Framework orienté-agent pour la modélisation et l'implantation de métaheuristiques Ce chapitre constitue la première partie de notre contribution. Il présente de manière approfondie le framework *AMF* en commençant par la description du modèle organisationnel de métaheuristique. Ensuite, nous détaillons le guide méthodologique. Nous terminons par une série d'exemples d'analyse de métaheuristique utilisant le framework *AMF*.

Chapitre 5 - Conception d'une métaheuristique à base de coalition d'agents Dans cette seconde partie de notre contribution, nous présentons *CBM* : une métaheuristique fondée sur la métaphore de la coalition. La conception de cette méthode s'appuie sur le framework *AMF*. Nous décrivons dans un premier temps le modèle de cette métaheuristique puis nous détaillons les mécanismes d'apprentissage et de coopération mis en œuvre.

Chapitre 6 - Application de *CBM* au problème de tournées de véhicules Nous considérons dans ce chapitre une première application de notre approche sur un problème de tournées de véhicules. Dans un premier temps, nous essayons de valider expérimentalement les mécanismes d'apprentissage et de coopération mis en oeuvre dans *CBM*. Ensuite, nous confrontons les résultats expérimentaux de *CBM* à différentes approches de résolution existantes.

Chapitre 7 - Application de *CBM* au problème de positionnement Ce chapitre propose une seconde application de *CBM* sur un problème de positionnement. En plus d'apporter une seconde validation expérimentale de l'efficacité de l'approche, nous cherchons à illustrer son aspect de flexibilité, c'est-à-dire, la facilité d'adaptation sur différents problèmes d'optimisation.

Chapitre 8 - Conclusion Le mémoire se termine par un bilan des travaux de recherche. Nous dressons un ensemble de perspectives pouvant être développées par la suite, ouvrant ainsi de nouvelles voies et propositions dans le but d'améliorer ce travail.

CHAPITRE 2

OPTIMISATION COMBINATOIRE ET MÉTAHEURISTIQUES

Sommaire

2.1 Introduction

L'objectif de ce chapitre est de situer et de comprendre les métaheuristiques, mais aussi de souligner l'intérêt d'apporter de nouveaux outils pour assister la conception et l'implantation de ces méthodes d'optimisation. Dans un premier temps, les termes liés à l'optimisation combinatoire et aux métaheuristiques sont définis. Puis nous donnons un aperçu des principales métaheuristiques. Malgré la diversité des approches existantes, nous nous efforçons dans cet état de l'art de faire ressortir les concepts communs des métaheuristiques. Ensuite, nous présentons plusieurs modèles et frameworks dont l'objectif est d'établir un cadre commun d'analyse et de conception des métaheuristiques. L'observation et la comparaison de ces différents modèles nous conduiront à exposer l'intérêt qu'il peut y avoir de proposer un nouveau framework suivant le paradigme des systèmes multi-agents.

2.2 Notions de base en optimisation combinatoire

Les problèmes que nous traitons dans le cadre de cette thèse relèvent de l'optimisation combinatoire. Un problème d'optimisation combinatoire, peut être formulé de la manière suivante :

Définition 2.1 Problème d'optimisation combinatoire, inspiré de [Sakarovitch, 1984]

Soit S l'ensemble des solutions possibles d'un problème. Soit $X \subseteq S$ l'ensemble des solutions admissibles vérifiant un ensemble de contraintes C. Soit $f : X \mapsto \mathbb{R}$, une fonction que l'on nomme fonction objectif. Un problème d'optimisation se définit par :

$$\begin{cases} \min f(x) \\ x \in X \end{cases}$$

Lorsque l'ensemble S des solutions est discret, on parle de problème d'optimisation combinatoire.

On considère indifféremment des problèmes de minimisation et des problèmes de maximisation. En effet, on pourra transformer un problème de maximisation en un problème de minimisation en considérant que : $\max f(x) = -\min(-f(x))$.

Ainsi, les éléments qui définissent un problème d'optimisation sont (i) une fonction objectif f, (ii) un ensemble de solutions S, et (iii) un ensemble de contraintes C que doit satisfaire une solution afin d'être admissible. Il existe cependant des variantes dans la définition des problèmes d'optimisation combinatoire. En effet, pour des problèmes tels que les problèmes d'optimisation multiobjectifs et les problèmes de satisfaction de contraintes, il est souvent utile de revoir la définition qui vient d'être fournie. Une présentation de ces deux types particuliers de problèmes d'optimisation est donnée respectivement dans [Collette and Siarry, 2002] et [Tsang, 1993].

Il est important de distinguer “problème” et “instance de problème”. La notion de “problème” se réfère à une question générale exprimée de manière générique. Cette question possède généralement des paramètres dont la valeur reste à fixer. Une instance d’un problème est obtenue en explicitant la valeur de chacun des paramètres du problème instancié. Par exemple la question : “Quelle est la route la plus courte permettant de desservir un ensemble de clients ?”, définit un problème d’optimisation, et lorsque l’on donne la valeur des positions de chacun des clients on décrit une instance de problème.

2.3 Approches heuristiques et métaheuristiques

Dans cette thèse, nous nous intéressons à la résolution de problèmes d’optimisation combinatoire par des méthodes heuristiques. Le texte suivant sert d’introduction pour illustrer l’idée de recherche heuristique.

> *Pour réaliser un mémoire, un étudiant doit faire quelques recherches bibliographiques. En plus des nombreuses sources d’informations classiques qu’il peut exploiter, il décide de se rendre dans une bibliothèque privée spécialisée dans son domaine. Le problème de cette bibliothèque est qu’elle ne propose aucun catalogue pour faciliter la recherche d’un livre. Il faut donc regarder directement dans les rayons pour effectuer une recherche, et pour se rendre compte de la pertinence d’un livre, il est nécessaire de le feuilleter. À la vue des centaines de livres dans la bibliothèque, notre étudiant ne peut pas se permettre de regarder chaque ouvrage pour emprunter celui qui lui convient le mieux. Une telle manière de faire prendrait plusieurs jours. Utilisant le fait que les livres ont une disposition par rayon relativement cohérente, il va mettre en place un procédé de recherche lui permettant de trouver un ouvrage satisfaisant en un temps raisonnable. La stratégie de recherche consiste à piocher aléatoirement un livre et le feuilleter. Si l’ouvrage semble intéressant, il va regarder les ouvrages proches jusqu’au moment ou ils semblent ne plus convenir. À ce moment-là, il réitérera le processus en piochant au hasard un autre livre, de préférence dans un rayon qu’il n’a pas encore exploré. Si au bout de plusieurs itérations il s’aperçoit que cette stratégie n’est pas efficace, alors il la modifiera. Lorsque notre étudiant souhaitera arrêter sa recherche, il empruntera le livre le plus pertinent parmi ceux qu’il a feuilletés.*

Dans cet exemple, le problème auquel fait face l’étudiant est un problème d’optimisation pouvant s’exprimer sous la forme suivante : considérant les livres présents dans la bibliothèque, trouver le plus approprié au sujet de recherche de l’étudiant. Dans ce problème d’optimisation, chaque livre est une solution potentielle et la fonction objectif à maximiser est la pertinence du livre.

Pour ce problème, l'étudiant ne dispose pas d'outils lui permettant de trouver en un temps raisonnable la meilleure solution. Il met donc en place une recherche de type heuristique. Le terme "heuristique" au sens originel qualifie "tous les outils intellectuels, tous les procédés et plus généralement toutes les démarches favorisant la découverte ou l'invention" [Chrétien-Goni, 2005]. Dans le cadre de la résolution de problèmes, les heuristiques sont définies comme étant des techniques de résolution incertaines dont l'usage n'est justifié que par le constat d'une efficacité antérieure dans des problèmes analogues [Chrétien-Goni, 2005]. L'emploi de méthodes de recherche heuristiques se justifie par l'inexistence ou l'inefficacité de méthodes exactes, et la difficulté ou l'impossibilité d'explorer exhaustivement l'espace des solutions.

Dans l'exemple, l'heuristique consiste à choisir aléatoirement un livre puis à effectuer une recherche localement en examinant les livres proches. Ce processus est réitéré jusqu'à un critère d'arrêt. Cette procédure qui utilise la proximité nous permet d'illustrer la notion de voisinage en optimisation combinatoire. Un voisinage permet de structurer l'espace de recherche en définissant pour chacune des solutions, un ensemble de solutions voisines à examiner. Le voisinage d'une solution peut être défini de la manière suivante :

Définition 2.2 Voisinage

Le voisinage d'une solution s est un sous-ensemble de solutions appartenant à S (l'ensemble des solutions possibles) atteignable à partir d'une transformation $\mathcal{N}(s)$. On dit qu'une solution s' est voisine d'une solution s, si s' appartient à $\mathcal{N}(s)$. L'application $\mathcal{N}$ est appelée structure de voisinage.

L'inconvénient d'utiliser une recherche locale, c'est-à-dire une méthode fondée sur un ou plusieurs voisinages, est le risque de tomber sur des solutions de mauvaise qualité qui ne peuvent pas être améliorées par un déplacement dans le voisinage. On dit que la recherche a atteint un minimum local. Le résultat risque alors de ne pas être satisfaisant puisque l'objectif est de trouver une solution proche de l'optimum global. Plus précisément, nous définissons ces deux types d'optimum de la manière suivante.

Définition 2.3 Optimum local, inspiré de [Hansen and Mladenović, 2003]

Dans le cas d'un problème de minimisation, une solution x est un optimum local pour la structure de voisinage $\mathcal{N}$, si il n'existe pas de solution $x' \in \mathcal{N}(x)$ telle que $f(x') < f(x)$. Pour un problème de maximisation, l'inégalité est inversée.

Définition 2.4 Optimum global, inspiré de [Hansen and Mladenović, 2003]

Un optimum global est un optimum local sur l'ensemble des structures de voisinages possibles. Ainsi, dans le cas d'un problème de minimisation un optimum global est une solution s telle qu'il n'existe pas de solution $s' \in S$ avec $f(x') < f(x)$. Pour un problème de maximisation, l'inégalité est inversée.

L'heuristique présentée dans l'exemple est spécifique à la recherche d'un livre dans une bibliothèque. Si l'on dégage de cette heuristique un schéma général applicable à d'autres problèmes d'optimisation, alors ce schéma sera qualifié de métaheuristique. Le terme métaheuristique a été introduit pour distinguer les approches spécifiques à un problème (les heuristiques) des méthodes génériques applicables à différents problèmes (les métaheuristiques). Dans [Blum and Roli, 2003], les auteurs résument les propriétés attachées à la notion de métaheuristique :

- Les métaheuristiques sont des stratégies qui "guident" le processus de recherche.
- L'objectif est d'explorer efficacement l'espace de recherche afin de trouver une solution proche de l'optimal.
- Les techniques mises en œuvre dans les métaheuristiques vont de la simple recherche locale à des procédures complexes d'apprentissage.
- Les métaheuristiques sont des algorithmes de résolution incertains et souvent non déterministes.
- Elles peuvent intégrer des mécanismes évitant d'être piégé dans une zone de l'espace de recherche.
- Les métaheuristiques sont décrites suivant un niveau d'abstraction indépendant du problème spécifique à traiter.
- Les métaheuristiques peuvent encapsuler les informations spécifiques au problème sous la forme de sous-heuristiques contrôlées à un niveau supérieur.
- Les métaheuristiques les plus modernes introduisent des mécanismes pour adapter et guider la recherche dynamiquement. On parle d'approches adaptatives et auto-adaptatives.

Les diverses métaheuristiques sont présentées sous des formes variées. Néanmoins, nous pensons que des principes communs et essentiels aux métaheuristiques sont exprimés par les concepts classiques d'intensification et de diversification, d'une part, et par les notions de stratégie de recherche et d'adaptation ou d'auto-adaptation de cette stratégie de recherche, d'autre part. À notre avis, ces concepts recouvrent les aspects essentiels que nous devrions retenir dans la conception de nouvelles métaheuristiques. Nous verrons comment ces éléments s'articulent les uns par rapport aux autres dans les différents schémas existants de métaheuristiques. Cette analyse des métaheuristiques, que nous retrouvons dans l'exemple de la bibliothèque, peut être résumée par les deux points suivants :

- Une métaheuristique combine des tendances d'intensification et de diversification. L'intensification est la concentration de la recherche dans les zones prometteuses tandis que la diversification vise à explorer de nouvelles zones de l'espace de recherche.
- Pour être plus efficace, une métaheuristique peut éventuellement modifier sa stratégie de recherche, pendant le cours même de la recherche. On parlera alors d'adaptation de la stratégie de recherche.

Dans notre exemple de la bibliothèque, l'intensification correspond à la recherche locale autour d'un premier choix de livre et la diversification correspond au choix aléatoire d'un nouveau livre permettant de démarrer une recherche locale. Enfin, l'adaptation correspond à un changement de stratégie si l'étudiant s'aperçoit que sa recherche n'est pas efficace. Cette adaptation peut consister, par exemple, à modifier le type de voisinage utilisé ou encore le mode de recherche dans le voisinage. Les possibilités sont variées.

2.4 Principales métaheuristiques

Dans cette section nous donnons un aperçu des principales métaheuristiques. Celles-ci sont regroupées selon trois critères : le nombre de structures de voisinage exploitées, le nombre de solutions utilisées simultanément et le type de mémoire guidant la recherche. Ces critères amènent à une classification réalisée en fin de section.

2.4.1 Métaheuristiques de trajectoire

La première classe de métaheuristiques que nous identifions regroupe les métaheuristiques de trajectoire. Elles consistent à faire évoluer une solution en utilisant comme outil de base une procédure de recherche locale. Cette procédure améliore une solution par des déplacements successifs dans un voisinage. Le problème rencontré lorsque l'on améliore une solution en utilisant un voisinage est l'obtention d'un optimum local pouvant correspondre à une solution de mauvaise qualité. Différentes stratégies sont développées afin de contourner ce problème.

Il est possible d'identifier au moins cinq stratégies différentes permettant la sortie d'un optimum local. Chacune d'elle correspond à un type particulier de métaheuristique de trajectoire. Le principe de ces métaheuristiques est représenté dans la figure 2.1 inspirée de [Blum and Roli, 2003]. Dans ces différents schémas, l'espace des solutions est représenté en une dimension. Cette dimension représente aussi le voisinage. La fonction de coût fait apparaître un optimum local et un optimum global indiqués dans le schéma de légende 2.1.a.

La première stratégie permettant la sortie d'un optimum local consiste à appliquer une perturbation à la solution une fois l'optimum local atteint. Ce principe est utilisé dans la **recherche locale itérée** (*Iterative Local Search, ILS*).

La recherche locale itérée consiste à alterner une recherche locale dans un voisinage donné, et une procédure de perturbation afin de s'en échapper. La perturbation consiste à effectuer une série de mouvements aléatoires pouvant dégrader la solution courante. Une fois qu'une solution perturbée est obtenue, la procédure de recherche locale est de nouveau appliquée. Cette recherche peut mener à un nouvel optimum local, comme à l'optimum local initial. Le processus est ensuite réitéré jusqu'à la vérification d'un critère d'arrêt. La

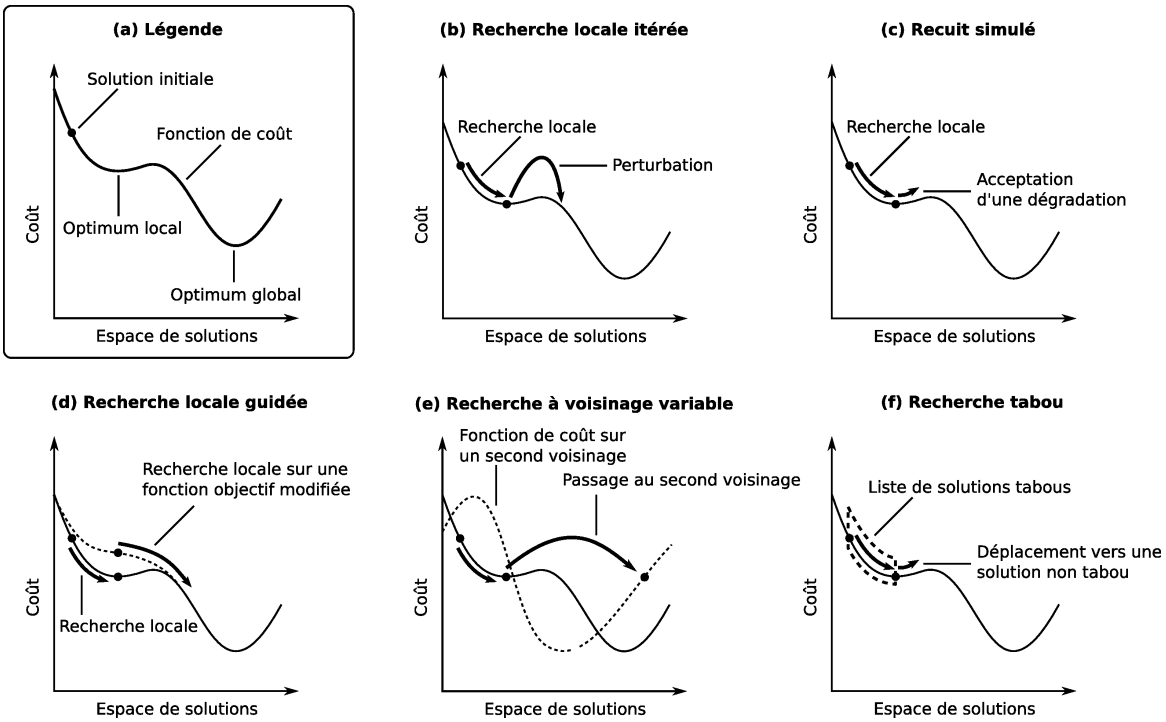

Figure 2.1: Principe de différentes métaheuristiques de trajectoire

figure 2.1.b illustre le principe de la recherche locale itérée. Une description détaillée est donnée dans [Lourenço et al., 2003].

Pour éviter de descendre trop rapidement dans un optimum local ou permettre d'en sortir, un deuxième type de stratégie est d'accepter des mouvements dégradant le coût de la solution au cours de la recherche locale. Ce principe est suivi dans la méthode du **recuit simulé** (*Simulated Annealing, SA*).

Le recuit simulé a fait son apparition en optimisation combinatoire en 1983 dans [Kirkpatrick et al., 1983]. Il s'agit d'une approche métaheuristique parmi les plus anciennes. Elle est fondée sur une analogie avec un phénomène de physique portant sur le refroidissement de matériaux. L'idée fondamentale est d'accepter des déplacements dans le voisinage d'une solution en dépit du fait qu'ils dégradent le coût de la solution et cela suivant une probabilité calculée à partir d'une température T du système à un moment donné. Ce principe est illustré par la figure 2.1.c. L'algorithme débute avec l'initialisation d'une solution courante s à une température T. À chaque itération une nouvelle solution s' est prise aléatoirement dans le voisinage de la solution courante. Cette nouvelle solution s' est acceptée comme solution courante si son coût est inférieur à celui de la solution courante ($f(s') < f(s)$ dans le cas d'un problème de minimisation) sinon l'acceptation de la solution dépend d'une probabilité calculée à partir de la température T et de la différence de coût $f(s') - f(s)$. Durant la recherche, la température T décroît, simulant un phénomène de refroidissement. Ainsi, au début la probabilité d'accepter une solution de moins bonne qualité est élevée, puis progressivement le processus devient une recherche locale classique acceptant uniquement des

mouvements améliorant la solution. Une présentation détaillée du recuit simulé est donnée dans [Henderson et al., 2003].

Une troisième stratégie permettant la sortie d'un optimum local consiste à modifier temporairement la fonction objectif du problème. Cette stratégie est employée par la **recherche locale guidée** (*Guided Local Search, GLS*).

Le principe de la recherche locale guidée, illustré dans la figure 2.1.d. consiste à modifier la fonction de coût de manière à rendre l'optimum local où se trouve la solution moins désirable. Pour cela la fonction de coût est modifiée avec un ensemble de pénalités. Lorsque la recherche locale a atteint un optimum, les pénalités les plus représentatives dans la solution courante sont augmentées et les autres sont diminuées. Ensuite, une nouvelle recherche locale est lancée en utilisant la fonction de coût modifiée. Un état de l'art de la recherche locale guidée et de ces applications est réalisé dans [Voudouris and Tsang, 2003].

D'une manière analogue à la recherche locale guidée, il est possible de changer la nature du voisinage lorsque la solution est bloquée dans un optimum local. Cette stratégie correspond à la **recherche à voisinage variable** (*Variable Neighborhood Search, VNS*).

La recherche à voisinage variable est une métaheuristique qui a été proposée par Hansen et Mladenović [Hansen and Mladenović, 2003]. La méthode consiste à changer dynamiquement le type de voisinage utilisé. Il existe différentes versions de recherche à voisinage variable. La méthode la plus simple consiste à choisir aléatoirement, à chaque itération, un type de voisinage puis à appliquer une recherche locale suivant ce voisinage jusqu'à obtenir un optimum local. Cette procédure est réitérée après une éventuelle perturbation. Ce principe est illustré dans le schéma de la figure 2.1.e. On remarque que le coût de la solution évaluée n'est pas modifié lors du passage d'un voisinage à un autre. Seul le paysage (*fitness landscape*) induit par le voisinage relativement à la fonction objectif est modifié.

En dernier lieu, une stratégie de recherche parmi les plus connues et dont le principe reste simple mais néanmoins efficace, consiste à empêcher le retour vers des solutions déjà visitées lors des déplacements dans le voisinage. Pour cela, une mémoire des dernières solutions visitées est entretenue et mise à jour pendant la recherche. Ce principe est utilisé dans le cadre de la **recherche tabou**, (*Tabu Search, TS*).

L'idée fondamentale de la recherche tabou est de conserver une trace des solutions récemment visitées dans une liste tabou. Cette liste est utilisée lors des déplacements dans le voisinage de manière à ne pas revenir sur des solutions déjà visitées. Pour cela, lors de la recherche locale, le voisinage de la solution courante est réduit aux solutions ne faisant pas partie de la liste tabou. Ce principe est illustré dans le schéma de la figure 2.1.f. Les déplacements réalisés consistent alors à sélectionner la solution de meilleur coût dans le voisinage. Il faut remarquer que celle-ci est choisie quand bien même elle possède un coût supérieur à la solution courante, de manière à pouvoir s'échapper des optima locaux. La liste tabou ne conserve que les solutions visitées les plus récentes. Elle est nommée en conséquence "mémoire à court terme". Pour améliorer l'efficacité de la recherche tabou, certains auteurs

ajoutent des mémoires à moyen et long termes. Le principe détaillé de la recherche tabou ainsi que plusieurs de ses variantes sont présentés dans [Gendreau, 2003] et [Battiti and Tecchiolli, 1994].

Les cinq approches présentées effectuent une recherche en partant d'une solution unique initiale. Celle-ci évolue ensuite suivant l'application d'un ou plusieurs types de voisinages, et selon des règles favorisant la sortie des minima locaux. Nous retrouvons ainsi de manière bien explicite des processus d'intensification et de diversification de la recherche. D'autres métaheuristiques vont en revanche introduire un aspect de multiplicité dans la recherche via l'utilisation d'une population de solutions. Ces métaheuristiques sont présentées dans la section suivante.

2.4.2 Métaheuristiques à base de population

Les **algorithmes évolutionnistes** (*evolutionary algorithm*) forment une classe importante des métaheuristiques à base de population. Ces approches sont fondées sur la métaphore de l'évolution et de la sélection naturelle. Le principe est de faire évoluer une population d'individus (solutions) en appliquant des procédures de croisement, mutation et sélection. Chaque nouvelle génération d'individus est obtenue en croisant et mutant les individus de la génération courante, puis en appliquant une sélection sur la population composée des nouveaux et des anciens individus. Le principe des approches évolutionnistes est illustré par l'algorithme 1. Dans cet algorithme, P représente la population courante et P' les nouveaux individus obtenus par croisement et mutation. La procédure de croisement consiste à créer de nouvelles solutions en combinant les composantes de solutions initiales. La mutation d'une solution engendre une solution dont les composantes ont été légèrement modifiées par rapport à la solution initiale. Enfin, la sélection consiste à choisir les meilleures solutions pour former la nouvelle génération. Ce choix est souvent réalisé de manière probabiliste en fonction du coût (ou *fitness*) des solutions.

Algorithme 1 : Principe des approches évolutionnistes

Générer la population initiale P
Évaluer les individus de P
tant que *le critère d'arrêt n'est pas atteint* **faire**
  Croiser les individus de P pour obtenir P'
  Muter les individus de P'
  Évaluer les individus de P'
  Sélectionner la nouvelle génération P à partir de P et P'
fin

On distingue classiquement trois catégories principales d'approches évolutionnistes : la programmation évolutionniste (*evolutionary programming*), les stratégies d'évolution (*evolution strategies*), et les algorithmes génétiques [Blum and Roli, 2003]. Une présentation

détaillée de ces trois catégories est donnée dans [Hoffmeister and Bäck, 1991] et [Bäck et al., 1993]. Nous donnons ci-dessous quelques éléments de comparaison tirés de [Bäck et al., 1997] :

Algorithmes génétiques : l'opérateur de croisement est considéré comme étant le plus important des opérateurs. La mutation est appliquée avec des probabilités très faibles et agit en tant qu'opérateur d'arrière-plan. Le codage des solutions consiste en une représentation généralement binaire des individus.

Stratégies d'évolution : le codage des solutions peut être réalisé par des structures de données plus complexes que dans les algorithmes génétiques. Par ailleurs, les opérateurs de mutation ont une place aussi importante que les opérateurs de croisement.

Programmation évolutionniste : elle est fondée essentiellement sur l'opérateur de mutation et n'utilise pas d'opérateur de croisement. Comme les Stratégies d'évolution, le codage des solutions peut faire intervenir des structures de données complexes. Cette approche a été développée initialement pour faire évoluer des automates à états finis.

D'autres métaheuristiques à base de population se distinguent des approches évolutionnistes, notamment en y ajoutant une procédure de recherche locale. C'est le cas des **algorithmes mémétiques** (*Memetic Algorithms, MA*) et de la **recherche dispersée** (*Scatter Search*).

Les algorithmes mémétiques s'inspirent non seulement du principe d'évolution d'une population, mais aussi de l'apprentissage individuel des membres de cette population durant leur existence [Krasnogor and Smith, 2005]. L'origine du nom de cette métaheuristique est le concept de "Mème", introduit par Dawkins. Un mème est un élément culturel, acquit par apprentissage et pouvant être retransmis de génération en génération [Moscato, 1989]. Concrètement, chaque individu fait évoluer une solution de manière indépendante par recherche locale, puis des opérateurs de croisement, mutation et sélection sont appliqués. De plus, des choix adaptatifs d'heuristiques sont parfois mis en œuvre dans certaines versions.

La recherche dispersée, est une métaheuristique qui se distingue des algorithmes évolutionnistes classiques par l'utilisation d'une procédure de recherche locale et par la généralisation de l'opérateur de croisement. Cette procédure de croisement peut être appliquée à un ensemble de solutions et pas uniquement à un couple de solutions [Glover et al., 2003].

Kennedy et Eberhart ont proposé dans [Kennedy and Eberhart, 1995] une métaheuristique à base de population fondée sur le principe de l'intelligence en essaims, l'**optimisation par essaims particulaires** (*Particle Swarm Optimization, PSO*).

Cette métaheuristique est parfois présentée à l'aide d'une métaphore apicole. Lorsqu'une abeille est chargée de chercher de la nourriture, elle va subir plusieurs influences et son comportement va obéir à plusieurs tendances. Une première tendance consiste à explorer l'espace proche lors du déplacement. Une deuxième tendance consiste à revenir sur le dernier endroit où elle a précédemment trouvé de la nourriture. Enfin, une troisième tendance consiste à essayer de tirer parti des informations en possession des autres abeilles sur

l'emplacement des sites prometteurs. Dans l'optimisation par essaims particulaires, le principe est similaire. Des particules se déplacent dans l'espace de solutions suivant un comportement inspiré des abeilles. Chaque particule possède une position courante associée à une solution. Elle se déplace en combinant trois composantes illustrées dans la figure 2.2. La première composante entraîne la particule vers un point accessible en tenant compte de sa vitesse propre. La seconde composante l'oriente vers la meilleure solution obtenue lors des précédents déplacements. La troisième composante entraîne la particule vers la meilleure solution connue qui a été mémorisée suite aux informations reçues des autres particules. La prochaine position de la particule est une combinaison de ces trois composantes. Ainsi, le déplacement des particules permet d'explorer l'espace de recherche, mais aussi de concentrer la recherche dans les zones prometteuses. Les mécanismes d'intensification et de diversification sont représentés et mis en œuvre par ces trois tendances. Cette approche est particulièrement adaptée aux problèmes d'optimisation en variables continues, mais plusieurs variantes utilisent une structure de voisinage pour définir l'espace de recherche dans le cadre de problèmes discrets [Kennedy and Eberhart, 1997]. Une présentation détaillée de l'optimisation par essaims particulaires et de ses variantes est donnée dans [Clerc, 2005].

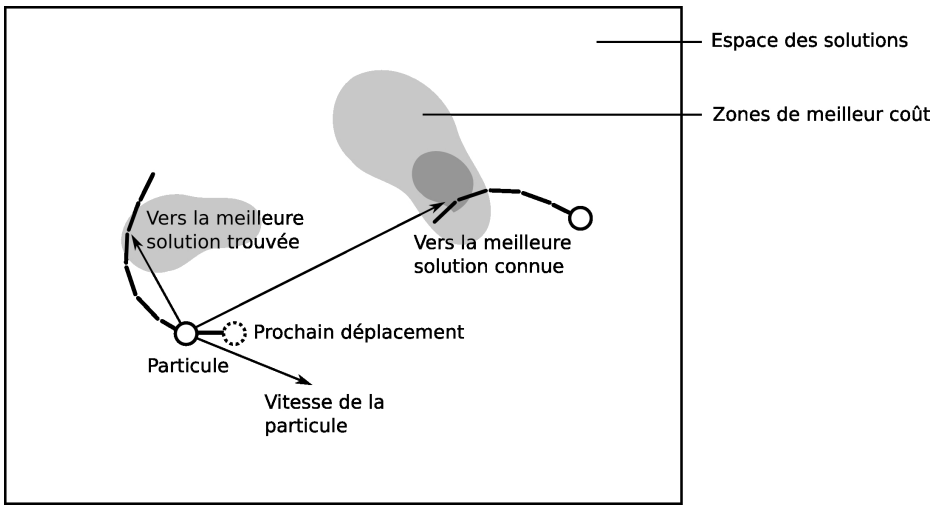

Figure 2.2: Principe du déplacement d'une particule dans l'optimisation par essaims particulaire

Toutes les métaheuristiques que nous venons de présenter utilisent une ou plusieurs solutions pour réaliser la recherche. Le processus itératif de chacune de ces métaheuristiques produit une ou plusieurs nouvelles solutions à partir des précédentes, en les modifiant (perturbation, recherche locale, etc.) ou en les combinant (opérateur de croisement). D'autres métaheuristiques génèrent de nouvelles solutions, non pas à partir des précédentes solutions, mais à partir d'informations réunies dans un modèle. On parle alors de métaheuristiques à base de modèle (*model-based metaheuristics*) [Branke et al., 2005, Zlochin et al., 2004]. Nous les présentons brièvement.

2.4.3 Métaheuristiques à base de modèle

Parmi les métaheuristiques à base de modèle, on trouve l'**optimisation par colonie de fourmis** (*Ant Colony Optimization, ACO*) et les **algorithmes à estimation de distribution** (*Estimation of Distribution Algorithms, EDA*).

L'optimisation par colonies de fourmis est une approche inspirée du comportement collectif des fourmis. Ce comportement permet aux fourmis de minimiser la longueur du chemin entre la source de nourriture et la fourmilière. Lorsque la fourmi suit un chemin, elle dépose une trace de phéromone qui est utilisée par les fourmis suivantes. Celles-ci ont d'autant plus tendance à choisir un chemin donné suivant que la concentration en phéromone y est forte. Il en résulte que leur action collective produit un chemin qui peut être le plus court. Dans l'algorithme d'optimisation par colonies de fourmis, la construction d'une nouvelle solution est effectuée par une fourmi. La mémoire qui guide cette recherche représente la matrice de phéromone. Celle-ci est utilisée par les fourmis pour construire de manière incrémentale les solutions. Chaque solution correspond à un ensemble de composantes. Lors du choix d'une composante de solution, la fourmi est soumise à une tendance à exploiter les informations de la matrice de phéromone, mais aussi à une tendance à utiliser des informations heuristiques [Dorigo and Stützle, 2000]. Ces deux tendances sont combinées dans le processus de décision stochastique de la fourmi. Après avoir créé une solution, un dépôt de phéromone dont l'intensité dépend de la qualité de la solution est effectué. Les dépôts successifs de phéromone combinés à l'évaporation vont conduire les fourmis à produire des solutions de meilleure qualité. Dans cette métaheuristique, la matrice de phéromone peut être vue comme un modèle probabiliste reflétant les caractéristiques des zones prometteuses de l'espace de recherche [Zlochin et al., 2004]. Dans [Maniezzo and Carbonaro, 2001] et [Dorigo et al., 1999], les auteurs présentent un état de l'art sur les variantes et les applications de l'optimisation par colonie de fourmis.

Les algorithmes à estimation de distribution consistent à estimer la distribution du coût des solutions en fonction des différentes composantes de solution ou variables de décision. La distribution doit faire apparaître les zones prometteuses de manière à y concentrer la recherche. La méthode débute avec un échantillonnage aléatoire de solutions. Une partie de ces solutions va permettre de faire une première estimation de la distribution de la fonction de coût. À partir de cette estimation, un nouvel échantillon de solutions est créé de manière probabiliste de façon à concentrer l'échantillon dans les zones prometteuses identifiées par l'estimation de distribution. Ce processus est réitéré afin d'affiner l'estimation de distribution et ainsi trouver de meilleures solutions. Un état de l'art sur les algorithmes à estimation de distribution est réalisé dans [Zlochin et al., 2004].

2.4.4 Classification des métaheuristiques

Dans les trois sections précédentes nous avons présenté un large panel de métaheuristiques. Le tableau 2.1 synthétise les caractéristiques de ces métaheuristiques et les classent à partir de trois critères : le nombre de solutions utilisées simultanément, le nombre minimum de structures de voisinage exploitées, et le type de mémoire guidant la recherche. Le premier critère permet de distinguer les métaheuristiques à base de population des métaheuristiques de trajectoire. Le second critère reflète l'usage de la recherche locale, et le dernier critère sépare les métaheuristiques à base de modèle des autres à base d'une mémoire de solutions. On retrouve ce type de classification dans plusieurs articles avec parfois des critères supplémentaires tels que la distinction entre les métaheuristiques d'inspiration naturelle et celles qui ne le sont pas [Blum and Roli, 2003]. Néanmoins, nous y retrouvons toujours ce souci d'intensification et de diversification dans la recherche. La stratégie employée guide la recherche, tantôt vers des zones précises, tantôt vers de nouvelles régions. Souvent entrelacées les deux tendances doivent être ajustées précisément par le paramétrage de l'algorithme.

2.5 Frameworks et librairies pour les métaheuristiques

Face à la grande variété de métaheuristiques, il semble nécessaire d'établir un cadre commun d'analyse et de conception. À partir de ce constat, plusieurs outils ont été proposés. Parmi ces outils, nous distinguons les frameworks pour la modélisation des métaheuristiques et les librairies visant à faciliter leur implantation.

Dans cette section, nous présentons tout d'abord les principaux frameworks proposés dans le domaine, puis nous donnons un aperçu de plusieurs librairies. Enfin, à partir d'une analyse des avantages et inconvénients des différentes approches, nous exposons l'intérêt qu'il peut y avoir de proposer un nouveau framework suivant le paradigme des systèmes multi-agents.

2.5.1 Principaux frameworks pour les métaheuristiques

Un framework peut être vu comme un ensemble d'outils facilitant le développement d'applications. Il fournit un modèle ou un squelette assez générique pour être réutilisé lors de la conception de différentes applications. Ainsi, chacun des frameworks présentés dans cette section est fondé sur un modèle permettant de décrire les différents composants des métaheuristiques. Nous présentons dans cette section quatre frameworks : *I&D Frame* (*Intensification and Diversification Frame*), *AMP* (*Adaptive Memory Programming*), *ALS* (*Adaptive Learning Search*) et *MAGMA* (*MultiAGent Metaheuristics Architecture*).

Métaheuristique	Nombre de solutions utilisées simultanément		Nombre minimum de structures de voisinage exploitées			Type de mémoire guidant la recherche	
	Mono-solution	Population	Aucune	Une	Plusieurs	A base de solution	A base de modèle
Recherche locale itérée	X			X		X	
Recuit simulé	X			X		X	
Recherche à voisinage variable	X				X	X	
Recherche tabou	X			X		X	
Recherche locale guidée	X			X		X	
Algorithme évolutionniste		X	X			X	
Algorithme mémétique		X		X		X	
Recherche dispersée		X		X		X	
Algorithme à estimation de distribution		X	X				X
Optimisation par colonie de fourmis		X	X				X
Optimisation par essaim particulaire		X		X		X	

Table 2.1: Classification des principales métaheuristiques

I&D Frame, (*Intensification and Diversification Frame*)

I&D Frame introduit par Blum et Roli [Blum and Roli, 2003] est un modèle de métaheuristiques mettant en avant les concepts d'intensification et de diversification. Dans cette approche les métaheuristiques sont analysées en termes de composants d'intensification/diversification (*I&D components*). Ces composants correspondent aux opérateurs, actions ou stratégies mis en place pour guider la recherche de solutions. Ils sont classés en trois catégories : composants guidés par la fonction objectif (notés *OG*, *Objective function Guided*), composants guidés par une ou plusieurs fonctions autres que la fonction objectif (notés

NOG, *Not Objective function Guided*) et composants non guidés ou aléatoires (notés *R*, *guided by Randomness*). Étant donné que les composants peuvent faire partie de plusieurs catégories, ils sont représentés sur un triangle dont les sommets correspondent aux caractéristiques *OG*, *NOG* et *R*. La figure 2.3 illustre cette représentation. Les trois catégories de composants sont associées à une des deux tendances d'intensification ou de diversification. Les composants faisant partie de la catégorie *OG* correspondent plus précisément à de l'intensification et les composants des catégories *NOG* et *R* se rapportent à de la diversification.

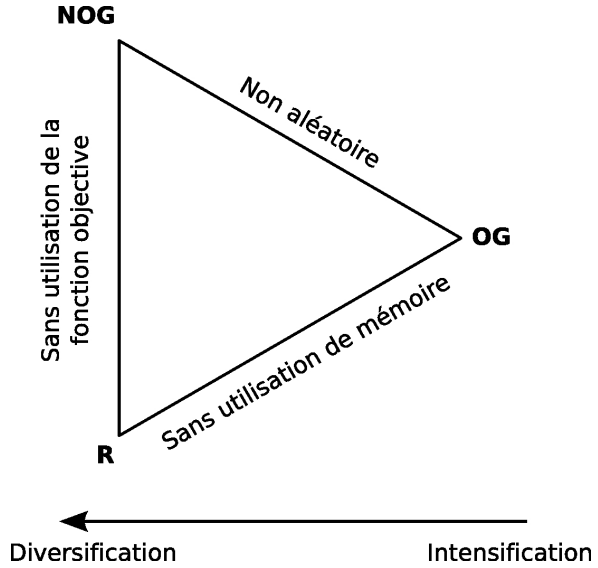

Figure 2.3: Classification des composants *I&D Frame*, d'après [Blum and Roli, 2003]

Afin d'illustrer ce modèle, prenons l'exemple d'une recherche tabou simple. L'élément clé de la recherche tabou est le choix de la solution dans le voisinage de la solution courante. Ce choix est effectué en fonction des critères tabous et de la valeur des solutions dans le voisinage. Ainsi, le composant effectuant le choix de la solution dans le voisinage de la solution courante est guidé par la fonction objectif (composant *OG*) et par la liste tabou qui elle-même n'est pas guidée par la fonction objectif (composant *NOG*). Le mécanisme de choix ainsi identifié peut être positionné sur le triangle représentant les trois catégories *OG*, *NOG* et *R*. Pour la recherche tabou, le composant "choix de la solution courante" sera positionné sur le segment *NOG-OG*. Ainsi, le schéma de la figure 2.4 représente une sorte de signature de la recherche tabou simple et permet d'analyser les mécanismes présents dans la métaheuristique. Ce composant combine les tendances d'intensification et de diversification. De plus, on remarque que la taille de la liste tabou va conditionner le rapport intensification/diversification.

L'approche *I&D Frame* permet ainsi d'analyser les composants de métaheuristiques qui guident la recherche. L'apport principal de cette approche est d'introduire deux critères permettant de caractériser les mécanismes au sein d'une métaheuristique : les tendances intensification/diversification et le type de fonction sur laquelle est fondé le mécanisme (*OG*, *NOG* ou *R*). Les auteurs de l'*I&D Frame* indiquent que ce type d'analyse doit permettre de

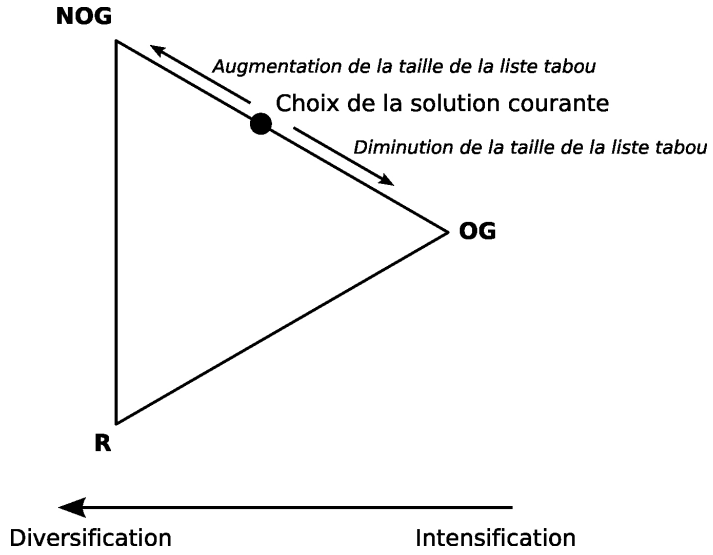

Figure 2.4: Composants *I&D Frame* pour la recherche tabou

faciliter la conception de métaheuristiques hybrides. Le modèle *I&D Frame* correspond à une première étape vers l'élaboration d'un framework permettant d'assister le développement de nouvelles métaheuristiques.

AMP, (*Adaptive Memory Programming*)

AMP est un modèle visant à unifier les concepts fondateurs des métaheuristiques. Le terme "*Adaptive Memory Programming*" a été introduit par Glover [Glover, 1997] pour définir la mémoire permettant de guider le processus de recherche dans les métaheuristiques et plus particulièrement dans la recherche tabou. Ce concept de mémoire adaptative a été étendu afin de produire un modèle unifié des métaheuristiques [Taillard et al., 2001]. Dans ce modèle, une métaheuristique est composée d'une mémoire adaptative et d'un processus itératif la faisant évoluer. Ce processus est décrit dans l'algorithme 2.

Algorithme 2 : Principe du processus itératif d'*AMP*, d'après [Taillard et al., 2001]

Initialiser la mémoire
tant que *le critère d'arrêt n'est pas vérifié* **faire**
  Générer une solution s à partir de la mémoire
  Améliorer la solution s par recherche locale, soit s' la solution obtenue
  Mettre a jour la mémoire avec s'
fin

L'approche *AMP* suppose qu'une métaheuristique est fondée sur une mémoire. Cette mémoire peut être de différentes natures. Elle peut être une solution unique (*SA, ILS, VNS*), un ensemble de solutions (*GA, MA*), une liste tabou (*TS*) ou encore une matrice de phéromone (*ACO*). Trois procédures sont utilisées pour effectuer la recherche : génération de solution à partir de la mémoire, amélioration de la solution, et mise à jour de la mémoire.

À partir de ces éléments, beaucoup de métaheuristiques peuvent être décrites.

En reprenant l'exemple de la recherche tabou, la mémoire adaptative correspond à une liste tabou. La génération d'une nouvelle solution est réalisée en choisissant dans le voisinage de la solution courante une solution ne violant pas les critères de la liste tabou. Enfin, la mise à jour de la mémoire consiste à rendre tabou le mouvement vers l'ancienne solution courante [Glover and Kochenberger, 2003].

L'approche *AMP* donne une vue unifiée des métaheuristiques fondée sur le concept de mémoire et sur le processus itératif précédemment décrit. Les auteurs du framework *AMP* [Taillard et al., 2001] décrivent à l'aide de leur modèle : les algorithmes génétiques, la recherche dispersée, l'optimisation par colonie de fourmis et la recherche tabou. Cette approche est de plus en plus utilisée pour présenter des métaheuristiques originales ou hybrides.

ALS, (*Adaptive Learning Search*)

L'approche *ALS* introduite dans [Dréo et al., 2007] est une extension d'*AMP*. Contrairement au modèle *AMP*, le principe algorithmique d'*ALS* fait ressortir une phase d'intensification et une phase de diversification. De plus, cette approche considère la mémoire comme un "échantillon", c'est-à-dire un ensemble d'informations sur la distribution des solutions. Le modèle de métaheuristique proposé dans cette approche est décrit par l'algorithme 3.

Algorithme 3 : Algorithme d'*ALS*, d'après [Dréo et al., 2007]

Initialiser l'échantillon
tant que *le critère d'arrêt n'est pas vérifié* **faire**
  Echantillonnage de manière explicite, implicite ou directe
  Apprentissage de manière à extraire des informations de l'échantillon
  Diversification de la recherche pour obtenir de nouvelles solutions
  Intensification de la recherche pour améliorer l'échantillon existant
  Remplacement du précédent échantillon avec le nouveau
fin

Cet algorithme de métaheuristique est principalement adapté aux approches à base de population. La phase d'échantillonnage permet d'obtenir un ensemble de solutions, éventuellement à partir d'une distribution. L'apprentissage consiste à extraire les informations pertinentes de l'échantillon précédemment obtenu. Les phases d'intensification et de diversification permettent de diriger la recherche respectivement vers les meilleures zones et de nouvelles zones de l'espace de recherche.

À partir de ce schéma de métaheuristique, les auteurs de l'approche *ALS* décrivent plusieurs métaheuristiques à base de population. Par exemple, pour les algorithmes évolutionnistes, l'échantillonnage est implicite et consiste à utiliser les individus précédemment sélectionnés. L'apprentissage correspond à l'application d'un opérateur de croisement, la diversification est réalisée par mutation et l'intensification est associée à la sélection.

L'intérêt majeur de l'approche *ALS* par rapport à *AMP* est de rendre compte aussi bien des métaheuristiques à base de modèle que des métaheuristiques à base de solution.

MAGMA, (*MultiAGent Metaheuristics Architecture*)

MAGMA est un framework introduit par Milano et Roli [Milano and Roli, 2004]. Les objectifs recherchés sont de pouvoir comparer les métaheuristiques existantes afin de faciliter leur hybridation et leur implantation. Dans cette approche, une métaheuristique est un système multi-agent composé de quatre niveaux. Chaque niveau dispose d'agents spécialisés dans le traitement d'une tâche générique. La figure 2.5 représente les différents niveaux identifiés dans l'approche *MAGMA*.

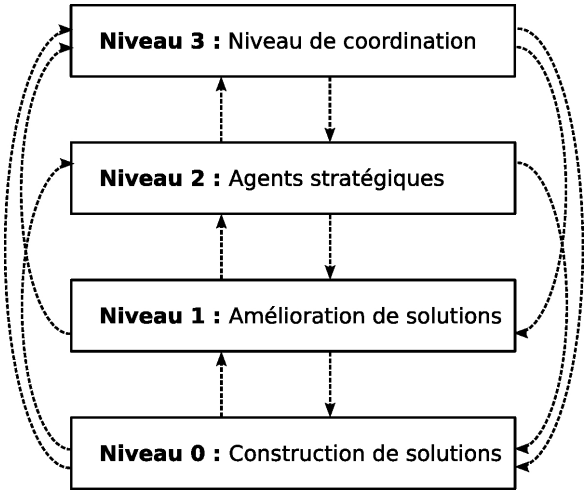

Figure 2.5: Architecture *MAGMA*, d'après [Milano and Roli, 2004]

Le niveau 0 est composé d'agents chargés de générer de nouvelles solutions pour les niveaux supérieurs. Le niveau 1 permet d'améliorer des solutions en utilisant des procédures telles que la recherche locale. Les agents du niveau 2 servent à contrôler le rapport diversification/intensification. Enfin, si plusieurs stratégies de recherche coexistent déjà dans le niveau 2, les agents de niveau 3 permettent de les coordonner.

L'approche *MAGMA* permet d'identifier différentes tâches au sein d'une métaheuristique. La notion d'agent est ici utilisée pour décrire un élément capable de réaliser une tâche particulière du système. On retrouve dans *MAGMA* les différentes procédures de l'approche *AMP*. De plus, l'approche agent prend en compte l'aspect de distribution de l'exécution.

2.5.2 Librairies pour les métaheuristiques

Les frameworks qui viennent d'être présentés ont pour objectif principal de faciliter l'analyse et la modélisation de métaheuristiques. Cependant, ces frameworks ne couvrent pas l'implantation de métaheuristiques. Pour cela, plusieurs librairies ont été développées indépendamment des frameworks précédemment proposés. Ces librairies ont pour objectifs

de simplifier l'implantation de métaheuristiques et de favoriser la réutilisation du code. Nous présentons ci-dessous les trois librairies de métaheuristiques qui nous semblent les plus représentatives et complètes.

***HotFrame* [Fink and Voss, 2002] :** La libraire *HotFrame*[1] (*Heuristic OpTimization FRAMEwork*) est un projet initié par Fink et Voss au sein de l'*Institute of Information Systems* de Hambourg. *HotFrame* est développée en *C++* et fournit un ensemble de classes relatives aux métaheuristiques et à leurs composants. Il s'agit d'une librairie "classique" dans le sens ou chaque métaheuristique est donnée sous la forme d'un squelette qu'il faut adapter au problème traité. Cette librairie couvre aussi bien les métaheuristiques de trajectoire que les métaheuristiques à base de population. Le but recherché par les auteurs de *HotFrame* est de combiner une implémentation efficace en termes de temps d'exécution avec la possibilité de réutiliser le code.

***ParadisEO* [Cahon et al., 2004] :** *ParadisEO*[2] est une librairie *Open Source* développée par l'INRIA de Lille. *ParadisEO* est une extension du framework *EO* (*Evolving Objects*) initialement conçu pour l'implantation d'algorithmes évolutionnistes [Cahon et al., 2004]. Les principaux objectifs de cette librairie sont, tout d'abord, de pouvoir réutiliser du code pour faciliter et accélérer le développement de métaheuristiques. Ensuite, l'architecture de la librairie a été conçue de manière à permettre le développement de nouvelles métaheuristiques ou de métaheuristiques hybrides. Enfin, *ParadisEO* fournit des solutions pour l'exécution distribuée des algorithmes. L'architecture de *ParadisEO* est composée de trois couches : *Solvers*, *Runners* et *Helpers*. Les *Helpers* forment la "couche basse" de la librairie qui fournit les composants de base des métaheuristiques tels que les opérateurs évolutionnistes (mutation, croisement, sélection) ou les méthodes de déplacement dans un voisinage. Les *Runners* correspondent aux différentes métaheuristiques définies à partir des *Helpers*. Les *Solvers* correspondent aux processus chargés d'exécuter et éventuellement de coordonner la ou les métaheuristiques. *ParadisEO* se distingue des autres librairies de métaheuristiques en donnant la possibilité de distribuer l'exécution. La distribution est envisagée au plus bas niveau, en distribuant par exemple l'exploration du voisinage d'une solution, ou à plus haut niveau en distribuant et coordonnant plusieurs processus de recherche.

***iOpt* [Dorne and Voudouris, 2004] :** *IOpt* (*Intelligent Optimisation Toolkit*) est un ensemble d'outils développés au sein de *BT Exact Technologies*, centre de recherche de *British Telecom*. Entièrement réalisé en *Java*, cet outil se compose d'une librairie pour définir les problèmes d'optimisation, d'une seconde librairie pour l'implantation de métaheuristiques, et d'un ensemble de logiciels de visualisation. La librairie relative aux problèmes d'optimisation est appelée *PMF* (*Problem Modelling Framework*). Elle permet de définir

[1] Site internet du projet *HotFrame* : `www1.uni-hamburg.de/IWI/hotframe/hotframe.html`
[2] Site internet du projet *ParadisEO* : `http://paradiseo.gforge.inria.fr`

un problème d'optimisation combinatoire à partir de classes ou d'interfaces représentant les variables de décision, les contraintes ainsi que la fonction objectif. La librairie de métaheuristiques est nommée *HSF* (*Heuristic Search Framework*). Elle fournit des composants de métaheuristiques (opérateurs évolutionnistes, déplacement dans un voisinage, etc.) ainsi qu'un ensemble de métaheuristiques prédéfinies. Cette librairie couvre les métaheuristiques de trajectoire et les métaheuristiques de population les plus répandues. Les outils de visualisation associés à *PMF* et *HSF* permettent, de visualiser les problèmes implémentés à partir de *PMF* et les composants d'un algorithme utilisant *HSF*, ou de suivre l'exécution d'un algorithme. La principale originalité de *iOpt* face aux autres librairies est de fournir des outils pour l'implémentation de problèmes et non pas seulement de fournir des squelettes de métaheuristiques.

2.5.3 Conclusion

Les frameworks qui ont été présentés décrivent une métaheuristique à l'aide d'un ensemble limité de concepts génériques. Le framework *I&D Frame* conçoit une métaheuristique sous la forme de composants agissant sur les tendances d'intensification et de diversification. *AMP* et *ALS* présentent une métaheuristique comme un processus itératif faisant évoluer une mémoire. *MAGMA* identifie au sein d'une métaheuristique quatre fonctions réparties dans une architecture en couche.

Bien que la terminologie et le vocabulaire utilisés pour décrire une métaheuristique soient différents d'un modèle à l'autre, il est possible d'identifier des concepts communs. Tout d'abord, les frameworks à l'exception d'*AMP* utilisent dans leur modèle les concepts d'intensification et de diversification. Dans *I&D Frame*, il s'agit de tendances liées aux composants stratégiques (*I&D components*). Dans *ALS*, la diversification et l'intensification sont associées à deux procédures permettant respectivement de chercher de nouvelles solutions et d'améliorer les solutions existantes. Dans *MAGMA*, les fonctions relatives aux niveaux 0 et 1 du modèle en couches correspondent à l'intensification et la diversification de la recherche. Ensuite, dans les différents frameworks, la mémoire sur laquelle est fondée la stratégie de recherche est un concept prédominant. Cette mémoire prend différentes formes selon la métaheuristique considérée. Par exemple, nous avons une liste de solutions dans la recherche tabou, une matrice de phéromone dans l'optimisation par colonie de fourmis, et une population de solutions dans les algorithmes évolutionnistes.

En mettant en perspective les apports des frameworks avec ceux des librairies de métaheuristiques, on s'aperçoit qu'il existe un écart important entre les modèles généraux proposés et les problématiques concrètes traitées par les librairies. Tout d'abord, les frameworks présentés à l'exception de *MAGMA* ne traitent pas de la potentialité de distribution de l'exécution des métaheuristiques. Dans *MAGMA* cet aspect est considéré en tant que relevant d'une distribution fonctionnelle. Chaque entité issue de la distribution correspond à la mise en œuvre d'une fonction précise dans le système. Le modèle ne semble pas considérer

l'aspect collectif de la résolution en tant qu'heuristique, mais davantage en tant que moyen de mise en œuvre d'une heuristique. Ensuite, on observe que ces frameworks n'intègrent pas les notions d'adaptation et d'auto-adaptation de la stratégie de recherche. L'adaptation dynamique de la stratégie de recherche, tout comme la potentialité de distribution liée à l'aspect de résolution collective, sont pourtant des éléments importants pris en compte dans les développements spécifiques de métaheuristiques [Crainic and Toulouse, 2003].

Enfin, il semble que les auteurs des frameworks n'abordent pas strictement l'aspect méthodologique qui permet de passer de la conception d'un modèle de métaheuristique à son implantation. En résumé, les frameworks présentés, introduisent des concepts intéressants pour l'analyse de métaheuristiques, mais ne semblent pas rendre compte suffisamment des articulations entre conception et implantation dans un contexte distribué.

Dans cette thèse, nous proposons un framework de métaheuristique permettant de remédier à certains inconvénients soulevés précédemment. Ce framework devra donc permettre d'envisager les métaheuristiques sous l'angle de leur mise en œuvre et de leur modélisation en tant que système distribué. Il devra intégrer des concepts liés à l'adaptation et à l'auto-adaptation dynamique de la stratégie de recherche. Il devra également apporter des éléments méthodologiques permettant de faire le lien entre l'analyse, la modélisation et l'implantation. Nous pensons que l'application du paradigme multi-agent à la conception de métaheuristiques peut apporter une réponse à ces attentes.

CHAPITRE 3

LES SYSTÈMES MULTI-AGENTS ET L'OPTIMISATION COMBINATOIRE

Sommaire

3.1 Introduction

Dans ce chapitre, nous essayons de mettre en évidence ce qui fait l'intérêt de l'approche agent utilisée dans le cadre de l'optimisation et des métaheuristiques. La première section pose les notions de base sur les systèmes multi-agents. Les sections suivantes dressent un état de l'art sur les approches agent en optimisation. Nous présentons tout d'abord, les méthodes les plus souvent référencées dans la littérature en tant que représentatives des applications des systèmes multi-agents dans le domaine de l'optimisation. Ensuite, nous étendons également la présentation vers des approches métaheuristiques partageant un bon nombre des caractéristiques essentielles des systèmes multi-agents. Nous concluons en défendant l'idée que la distribution, l'hybridation et l'intégration de mécanismes d'adaptation dans les métaheuristiques peuvent être encouragées par l'adoption de l'approche agent.

3.2 Notions de base sur les systèmes multi-agents

3.2.1 Définition de la notion d'agent et de système multi-agent

Parmi les nombreuses définitions du terme "agent", certaines semblent faire l'objet d'un consensus au sein de la communauté multi-agent. Pour définir un agent, Ferber [Ferber, 1995] relève certaines propriétés clés s'appliquant aussi bien aux systèmes naturels qu'artificiels.

On appelle agent une entité physique ou virtuelle

a) qui est capable d'agir dans un environnement,

b) qui peut communiquer directement avec d'autres agents,

c) qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),

d) qui possède des ressources propres,

e) qui est capable de percevoir (mais de manière limitée) son environnement,

f) qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),

g) qui possède des compétences et offre des services,

h) qui peut éventuellement se reproduire,

i) dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Cette définition insiste sur la capacité d'agir des agents, et non pas seulement de raisonner. L'action est d'après Ferber, un concept fondamental pour les systèmes multi-agents. Il repose sur le fait que les agents accomplissent des actions qui vont modifier l'environnement et donc leurs prises de décision futures. Une autre propriété essentielle des agents est l'autonomie. Les agents ne sont pas dirigés par un utilisateur ou un autre agent, mais par

un ensemble de tendances qui leur sont propres. Ces dernières peuvent prendre la forme de buts individuels à satisfaire ou de fonctions de satisfaction ou de survie que l'agent cherche à optimiser. Cette autonomie n'est pas seulement comportementale, mais porte aussi sur les ressources (énergie, CPU, quantité de mémoire, accès à certaines sources d'informations, etc.). Paradoxalement, ces ressources sont à la fois ce qui rend l'agent dépendant de son environnement, mais aussi, en étant capable de gérer ces ressources, ce qui lui donne une certaine indépendance vis-à-vis de lui.

Dans la définition proposée par Jennings, Sycara et Wooldridge [Jennings et al., 1998] de la notion d'agent, les auteurs s'intéressent essentiellement au domaine de l'informatique, c'est-à-dire aux agents mis en œuvre par des programmes et leur support matériel.

Définition 3.1 Agent, d'après [Jennings et al., 1998]

Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu.

Ici, les concepts clés permettant de définir un agent sont : le fait d'être situé, l'autonomie et la flexibilité. Le fait d'être situé dans un environnement exprime que l'agent est capable d'agir sur son environnement à partir d'entrées sensorielles reçues de celui-ci. L'autonomie d'un agent correspond à sa capacité à agir sans intervention directe d'un humain ou d'un autre agent, et sous-entend que l'agent a le contrôle de ses propres actions et de son état interne. Enfin, la flexibilité d'un système est définie selon [Wooldridge and Jennings, 1995] par les deux points suivants :

- *la réactivité :* un agent doit être capable de percevoir son environnement et de répondre dans les temps requis à un changement intervenant dans cet environnement.
- *la pro-activité :* les agents ne doivent pas simplement agir en réponse à leur environnement, mais doivent aussi prendre des initiatives appropriées en fonction de leurs objectifs.
- *la sociabilité :* un agent doit être capable d'interagir avec les autres agents quand la situation l'exige afin de réaliser ses tâches.

À partir de ces deux définitions de la notion d'agent, nous considérons un système multi-agent comme étant un ensemble d'agents partageant un environnement commun. Ces agents communiquent et collaborent pour achever des objectifs personnels ou collectifs. L'environnement peut être considéré, entre autre, comme l'espace partagé par les agents qui constitue le support de la communication.

L'intérêt de l'approche agent réside dans la possibilité de réalisation collective d'une tâche : les interactions entre agents expliquent le résultat obtenu globalement par le système multi-agent. On parlera alors d'intelligence collective. D'après Drogoul [Drogoul, 2005], cette notion d'intelligence que l'on attribue aux agents ou au système dans son ensemble se manifeste pour un observateur suivant les trois propriétés suivantes :

- *l'adaptabilité :* indique la capacité du système à adapter son fonctionnement à l'environnement et au contexte dans lequel il se trouve.
- *la variabilité :* implique que les solutions d'adaptation choisies ne soient pas stéréotypées, et soient surtout capables d'évoluer au cours du temps (l'apprentissage est, par exemple, une forme de variabilité dans l'adaptation).
- *l'intentionnalité :* ce n'est pas une qualité intrinsèque du système considéré, mais une qualité attribuée par un observateur, a priori ou a posteriori.

L'intentionnalité est la propriété de l'esprit par laquelle il a des buts, des croyances, des intentions et toutes sortes d'états mentaux dirigés vers des objets du monde. L'auteur indique ici que cette intentionnalité est attribuée au système par un observateur extérieur. On dit que celui-ci adopte une posture intentionnelle vis-à-vis du système [Wooldridge, 1992].

3.2.2 Architecture cognitive et réactive

Une question fondamentale soulevée dans la conception de SMA est : "Faut-il concevoir les agents comme des entités déjà intelligentes, c'est-à-dire capables de résoudre certains problèmes par eux-mêmes, ou bien faut-il les assimiler à des êtres très simples réagissant directement aux modifications de l'environnement ?" [Ferber, 1995]. Ces deux alternatives correspondent à deux écoles de pensée : cognitive et réactive.

Dans l'approche cognitive, chaque agent dispose des connaissances permettant de réaliser les tâches qui lui sont associées et de gérer les interactions avec les autres agents. On parle alors d'agents intentionnels, c'est-à-dire que l'on considère les agents comme étant autonomes et capables de planifier des actions afin d'atteindre les objectifs pour lesquels ils ont été conçus. Dans le même ordre d'idées, l'une des caractéristiques essentielles des agents cognitifs est la capacité de représentation (de soi, des autres, du groupe, de la tâche) qui permet aux agents de raisonner sur leur fonctionnement collectif [Drogoul, 2005]. De plus, les interactions entre agents prennent la forme de communication dans un langage spécifique. L'architecture d'agent qui semble être la plus représentative de l'approche cognitive est l'architecture BDI (Belief, Desir, Intention) [Rao and Georgeff, 1995]. Cette architecture, comme son nom l'indique, est fondée sur les notions d'attitudes mentales que sont la croyance, le désir et l'intention. Les croyances sont les éléments connus de l'agent sur son environnement. Les désirs se rapportent aux états de l'environnement que l'agent souhaiterait voir se réaliser. Les intentions correspondent aux actions que l'agent planifie pour satisfaire ses désirs.

Si dans l'approche cognitive l'intelligence est attribuée aux agents individuels, l'approche réactive soutient que l'intelligence est une propriété globale du système multi-agent. Celle-ci est obtenue par les comportements d'agents simples en interaction. Les agents réactifs ont une complexité qui ne va pas au-delà de celle d'un automate. Ils n'ont pas de représentation de l'environnement ni des autres agents, et souvent peu de mémoire, leurs

actions et leurs perceptions sont purement locales [Drogoul, 2005]. Dans ce cadre, c'est la multiplicité des interactions et leur caractère stochastique qui permettent d'obtenir, par auto-organisation et émergence de structures, des propriétés collectives robustes et adaptatives. Un exemple de cette approche est l'optimisation par colonie de fourmis. Dans ce système, chaque fourmi a un comportement simple, mais les interactions entre fourmis, réalisées par l'intermédiaire de la phéromone, permettent de produire progressivement une bonne solution.

Dans les faits, dans beaucoup d'approches multi-agents pour la résolution de problèmes, la distinction entre agent réactif et cognitif n'est pas aussi nettement tranchée. Des comportements de type réactif et cognitif peuvent être combinés. Nous verrons dans la suite de la thèse, lors de la proposition de *CBM*, que la métaheuristique que nous proposons exploite des aspects des deux types d'approches.

3.3 Les heuristiques orientées-agents

Parmi les approches agents pour l'optimisation, il est possible de distinguer les heuristiques orientées-agents des métaheuristiques orientées-agents. Les heuristiques orientées-agents exploitent la structure du problème alors que les métaheuristiques orientées-agents exploitent la structure de l'espace des solutions. Cela signifie que dans le premier type d'approche, chaque agent est associé à une partie du problème, les actions combinées des agents produisant alors une solution lorsqu'on les observe comme un tout. Dans le second type d'approche, il s'agit du processus de résolution qui est distribué. Ainsi, un agent est associé à une ou plusieurs solutions. Nous donnons ici un aperçu des travaux sur les heuristiques, car ils constituent une part importante des approches agents en optimisation. Trois exemples représentatifs de cette démarche sont donnés. Il est à noter que nos travaux se situent au niveau des métaheuristiques.

3.3.1 Eco-résolution et problèmes de satisfaction de contraintes

L'éco-résolution (*Eco-problem solving*) [Ferber and Jacopin, 1991] est une méthode de résolution distribuée de problèmes. Dans cette approche, au lieu de considérer le problème dans sa globalité, il est décomposé et réparti au travers de différents agents réactifs indépendants. Ainsi, un sous-objectif est affecté à chaque agent de manière à ce que le problème soit résolu si les sous-objectifs sont atteints. Pour atteindre son objectif, un agent est guidé par les deux règles suivantes : (i) le désir d'être satisfait et la possibilité d'agresser d'autres agents pour l'être, et (ii) la nécessité de fuir face aux agressions des autres agents. À partir de ces règles simples, le système doit évoluer globalement vers une satisfaction des sous-objectifs et par conséquent vers un état stable qui représente une solution au problème.

Plusieurs travaux ont utilisé l'éco-résolution pour traiter des problèmes d'optimisation.

Par exemple, dans [Ghédira, 1994], l'auteur propose une approche combinant le recuit-simulé et l'éco-résolution pour la résolution d'un problème de satisfaction de contraintes. Dans cette approche, un agent est associé à un sous-ensemble de variables dont il essaie d'instancier les valeurs afin de satisfaire des contraintes du problème qui lui sont également associées. Par leur caractère composite, les problèmes de satisfaction de contraintes sont de bons candidats à la modélisation distribuée. Par exemple, diverses approches proposent des versions distribuées asynchrones des algorithmes traditionnels de "baktracking" [Yokoo et al., 1998]. Dans ce cas, le mode de communication entre agents est de type asynchrone par échange de messages, sans mémoire partagée. La synchronisation des échanges est gérée directement par les agents sans contrôle centralisé. Plusieurs autres approches multi-agents ont été développées dans le cadre de la résolution distribuée de problèmes de satisfaction de contraintes [Piechowiak and Hamadi, 2002]. Les applications sont nombreuses, allant des problèmes d'affectation de fréquences [Picard et al., 2007] au contrôle manufacturier [Clair et al., 2008].

3.3.2 Réseau contractuel et approches de marché

D'autres approches heuristiques appliquées à des problèmes d'optimisation sont les approches à base de réseau contractuel (*contract net protocol*) [Smith, 1980]. Ici les agents servent généralement à représenter les entités et dispositifs matériels du problème lui-même. Dans un problème de tournées de véhicules par exemple, ces entités peuvent être les véhicules, les clients ou la compagnie de transport. Les problèmes traités sont alors naturellement considérés dans un contexte dynamique, stochastique, et temps-réel. La demande des utilisateurs varie en temps-réel, et le système doit s'adapter continuellement à cette demande changeante et fluctuante [Bachem et al., 1994, Bürckert et al., 1998]. Ce type d'approche peut-être enrichie par des métaphores de marché comme dans [Kozlak et al., 2004]. Le manager (la compagnie) propose des offres de transports (demandes des clients) aux contractants (les véhicules) qui répondent par une offre de prix d'achat. Le manager décide alors qui sera l'acheteur. Suivant leur degré de satisfaction, et pour optimiser leurs trajets, les véhicules peuvent éventuellement remettre en vente des offres de transports déjà achetées.

3.3.3 Champs de potentiels artificiels pour la résolution de problèmes de positionnement

Dans le cadre des problèmes de positionnement mono et multi-niveaux, Moujahed et al. [Moujahed et al., 2006] ont développé une heuristique multi-agent visant à exploiter l'aspect spatial du problème. Le positionnement est un problème riche et complexe qui consiste à optimiser le placement de ressources par rapport à un ensemble de demandes. Pour résoudre le problème, les auteurs proposent une heuristique basée sur un modèle d'in-

teractions inspiré des champs de potentiels. Chaque ressource est un agent subissant un ensemble de forces d'attraction et de répulsion. Ces forces amènent les agents à modifier leur position et entraînent globalement le système vers une configuration plus stable. De la même manière que pour l'éco-résolution, l'état stable obtenu représente une solution du problème. La méthode cherche à exploiter des phénomènes d'auto-organisation et d'émergence. Les avantages attendus sont alors la robustesse et la flexibilité, c'est-à-dire la tolérance aux modifications portant sur l'instance du problème et la possibilité d'application dans un contexte dynamique et stochastique. L'approche a été appliquée avec succès à un problème de positionnement d'arrêts de bus et à un problème de patrouille [Moujahed, 2007].

3.3.4 Avantages et limites des heuristiques multi-agents

Les systèmes multi-agents présentés dans les trois sous-sections précédentes comportent pour la plupart des agents réactifs simples dont les interactions favorisent l'émergence d'une solution au problème traité. Il existe une multitude d'heuristiques multi-agents fondées sur ce principe. Dès lors que l'on peut décomposer un problème en sous-objectifs associés à des agents, et que l'on peut définir les comportements et interactions entre agents permettant d'atteindre ces sous-objectifs, alors une heuristique multi-agent peut être envisagée.

L'intérêt de ces approches réside, en premier lieu, dans leur simplicité. En effet, la décomposition du problème est souvent naturelle et les règles qui régissent le comportement des agents sont relativement simples. Par exemple, dans l'heuristique pour la résolution du problème de positionnement [Moujahed et al., 2006], les agents sont associés aux ressources. Cette décomposition repose sur la distribution naturelle des composants spatialisés du problème. Le déplacement des agents est régi par des combinaisons de forces obtenues suivant un calcul relativement simple. Cette simplicité explique la flexibilité de ces approches, c'est-à-dire, leur potentialité à être adaptées lors de l'ajout de nouvelles contraintes sur l'espace de recherche. Enfin, les heuristiques multi-agents sont généralement robustes, dans le sens où le système est capable de s'adapter aux changements dynamiques pouvant intervenir sur l'instance du problème. Cependant, l'efficacité en terme de temps de calcul s'obtient souvent au détriment de la qualité de la solution obtenue. Dans ces heuristiques, le comportement des agents ne permet pas forcément de sortir des optima locaux et le résultat est généralement dépendant de la configuration initiale des agents. Pour remédier à ce problème, il est alors nécessaire d'exécuter plusieurs fois l'heuristique, ou de mettre en place une procédure de perturbation. Enfin, ces heuristiques sont spécifiques aux types de problèmes traités étant donné que les agents sont associés aux composantes spécifiques du problème.

3.4 L'approche agent et les métaheuristiques

Certaines métaheuristiques sont originellement présentées suivant le point de vue de la résolution collective de problèmes. On peut alors les qualifier de métaheuristiques multi-agents. C'est le cas par exemple de l'optimisation par colonies de fourmis ou de l'optimisation par essaims particulaires. Cependant, des aspects essentiels des systèmes multi-agents peuvent être observés dans d'autres métaheuristiques, et plus particulièrement dans les métaheuristiques distribuées et les métaheuristiques hybrides. Dans les deux sections suivantes, nous présentons des métaheuristiques que nous qualifions d'orientées-agents car présentées comme représentatives dans le domaine des systèmes multi-agents, puis nous nous efforçons d'identifier les aspects relevant des systèmes multi-agents présents à divers degrés dans d'autres métaheuristiques.

3.4.1 Approches de résolution collective

Les métaheuristiques telles que l'optimisation par colonie de fourmis et l'optimisation par essaims particulaires peuvent être décrites comme relevant du domaine des systèmes multi-agents. En effet, ces deux approches sont fondées sur une métaphore d'insectes sociaux ce qui permet de les décrire en termes d'agents et d'interactions.

Dans ces approches, un agent est associé à la construction ou à l'amélioration d'une solution unique. Les interactions entre agents visent à échanger des informations à propos des zones prometteuses de l'espace de recherche, de manière à rendre l'exploration plus efficace. Cette coopération entre agents est mise en œuvre de deux manières différentes. Dans l'optimisation par colonies de fourmis, les agents coopèrent par le biais de la matrice de phéromone. Cette situation particulière de coopération entre agents interagissant de manière indirecte se nomme la stigmergie [Parunak et al., 2003]. Dans l'optimisation par essaims particulaires, les agents interagissent directement d'égal-à-égal. Il s'agit dans ce cas de coordination [Parunak et al., 2003].

L'intérêt de l'approche agent tel qu'on peut le voir dans ces deux métaheuristiques est, tout d'abord, de pouvoir décrire de manière naturelle leur fonctionnement ce qui en facilite la conception. Ensuite, la distribution du calcul doit permettre une exploration plus efficace de l'espace de recherche tout en permettant une mise en œuvre parallèle possible de l'algorithme. Cependant, nous devons remarquer s'agissant de l'optimisation par colonies de fourmis que celle-ci suppose, dans sa version originale, un environnement partagé dans lequel évoluent les agents [Crainic and Toulouse, 2003]. Ce qui rend problématique sa mise en œuvre dans un contexte asynchrone sans mémoire partagée.

3.4.2 Métaheuristiques distribuées

Un premier aspect des métaheuristiques qui les rapprochent des systèmes multi-agents est la distribution du calcul. Celle-ci peut intervenir à différents niveaux d'une métaheuristique. Ainsi, dans [Crainic and Toulouse, 2003], les auteurs distinguent trois niveaux de distribution dans les métaheuristiques. Le premier niveau, ou "bas niveau" consiste à exécuter en parallèle des opérations indépendantes dans une même itération de la métaheuristique. Par exemple, il est parfois possible d'effectuer une évaluation parallèle des solutions, ou encore il peut être possible d'appliquer des opérateurs de voisinage, de mutation ou de croisement, sur différentes solutions en parallèle. Ce niveau de parallélisme a uniquement pour but de permettre d'améliorer le temps d'exécution sans pour autant modifier la qualité du résultat par rapport à une version séquentielle du même algorithme. Dans la suite, nous ne nous intéresserons pas à ce type de distribution car il semble peu adapté à une interprétation multi-agent. Le second niveau de distribution consiste à répartir la recherche dans l'espace des solutions sur plusieurs processus de recherche. Cette seconde stratégie de distribution du calcul est généralement implantée sous la forme d'une architecture maître/esclave. Le processus maître est chargé de partitionner l'espace de recherche et les processus esclaves explorent de manière indépendante la partie qui leur a été assignée. Le dernier niveau de distribution correspond à la mise en concurrence de plusieurs processus de recherche heuristique. Un exemple répandu de ce type de distribution de l'exécution est l'approche des îles pour les algorithmes évolutionnistes [Tanese, 1989]. Cette approche de distribution consiste à former un ensemble de sous-populations séparées les unes des autres et n'interagissant entre elles que périodiquement au travers de migrations d'individus. Chaque sous-population est gérée par un algorithme évolutionniste indépendant ayant éventuellement un paramétrage propre [Skolicki, 2005]. Ce type de mise en œuvre distribuée permet, d'une part, d'améliorer l'exploration de l'espace de recherche, et d'autre part, de laisser ouverte la possibilité de mettre en concurrence différentes stratégies de recherche ou différents paramétrages de l'algorithme.

Le concept d'agent est parfois explicitement mentionné dans les métaheuristiques distribuées. C'est le cas dans la méthode Co-search [Talbi and Bachelet, 2004]. L'architecture de Co-Search est donnée dans la figure 3.1. Le principe est d'utiliser plusieurs agents spécialisés pour faire évoluer un ensemble de solutions réunies dans une mémoire. Celle-ci se compose d'une partie relative aux zones de l'espace de recherche explorées et d'une partie correspondant aux zones de recherche prometteuses. Trois agents utilisent et mettent à jour cette mémoire. Le premier agent est nommé *diversifying agent*. Il est chargé de la diversification et fournit de nouvelles solutions dans les zones non explorées de l'espace de recherche, en exploitant les informations concernant les zones explorées. Le second agent, *intensifiying agent*, a pour objectif de fournir des solutions prometteuses. Le dernier agent, *search agent*, effectue une recherche à partir des solutions fournies par les deux agents précédents. L'heuristique développée dans [Talbi and Bachelet, 2004] utilise une recherche

tabou pour le *search agent*, un algorithme génétique pour le *diversifying agent* et un opérateur de perturbation pour l'*intensifiying agent*. L'objectif recherché dans Co-search est de faire coopérer plusieurs stratégies de recherche complémentaires.

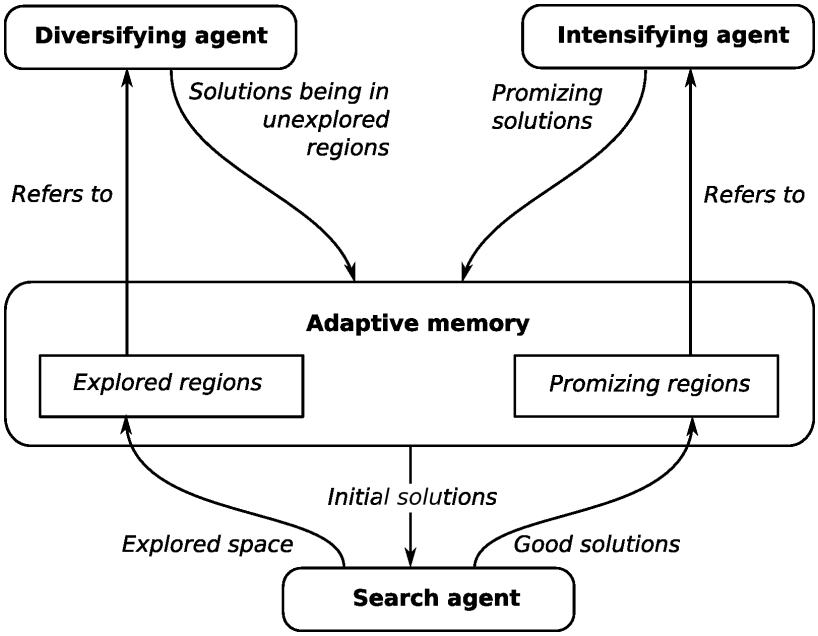

Figure 3.1: Principe de la métaheuristique *Co-search*, tiré de [Talbi and Bachelet, 2004]

3.4.3 Approches hybrides

Les algorithmes mémétiques [Merz, 2000] proposent une approche distribuée et hybride donnant lieu à des implantations très performantes. Un algorithme mémétique est une extension des algorithmes évolutionnistes habituels par hybridation avec une recherche locale. La métaphore qui en est donnée, par exemple dans [Merz, 2000], est celle de l'évolution culturelle résultant de l'apprentissage réalisé par des agents conscients qui retransmettent leur savoir de génération en génération. L'exécution de recherches locales indépendantes sur une population de solutions représente cet apprentissage. Des opérations de mutation, de croisement et de sélection complètent la dynamique collective. Il est à noter que les communications entre agents représentent une part minime du calcul et qu'il n'y a pas de mémoire partagée.

Une approche hybride combinant un algorithme mémétique avec une approche auto-organisée est présentée dans [Créput and Koukam, 2008]. Elle est appliquée spécifiquement à des problèmes d'optimisation spatialisés, dont plusieurs variantes d'un problème de tournées de véhicules combiné à un problème de positionnement. Ce problème consiste à définir un ensemble de routes de durée minimum permettant de desservir un ensemble de clients via des points de ramassage correspondant à des regroupements de clients. Dans cette approche, les clients et les points de passage des routes sont considérés comme des

agents en interaction et leur dynamique est mise en œuvre par l'algorithme d'apprentissage des cartes auto-organisatrices de Kohonen [Kohonen, 2001]. Afin de rendre l'heuristique plus compétitive, l'algorithme d'apprentissage joue le rôle d'un opérateur de recherche locale dans un algorithme mémétique. Ainsi, nous retrouvons des mécanismes de résolution distribués à deux niveaux. Le premier niveau est celui de l'approche mémétique opérant sur des populations de solutions, tandis que le deuxième niveau est celui de l'auto-organisation opérant au niveau des composants définissant le problème.

3.4.4 Métaheuristiques et apprentissage

Un autre aspect important associant les métaheuristiques aux systèmes multi-agents est lié aux problématiques d'adaptation et d'auto-adaptation de la recherche. En effet, le choix d'une bonne stratégie de recherche ou le choix d'un bon paramétrage peut être réalisé dynamiquement et cela peut être vu sous la forme d'un problème d'apprentissage [Birattari, 2005]. Les problématiques d'adaptation et d'auto-adaptation des choix d'opérateurs sont abordées dans la littérature sous des formes diverses. Par exemple, nous renvoyons à [Ong et al., 2006] pour un panorama sur les algorithmes mémétiques adaptatifs. Plus près des systèmes multi-agents, cette problématique est aussi abordée dans le cadre des hyper-heuristiques où l'adaptation de la recherche est assimilée à un problème d'apprentissage par renforcement [Burke et al., 2003]. Le problème de l'apprentissage par renforcement correspond au problème d'un agent qui doit choisir les actions à réaliser en fonction de la situation, de façon à maximiser son gain [Sutton and Barto, 1998]. Dans les hyper-heuristiques, le problème se situe au niveau du choix des heuristiques ou opérateurs les plus efficaces en temps d'exécution et en qualité de solution. Nous donnons ci-après une description des hyper-heuristiques.

Les hyper-heuristiques peuvent être définies comme étant des méthodes d'optimisation ayant la particularité d'utiliser une approche heuristique pour sélectionner les heuristiques qui vont résoudre le problème d'optimisation [Burke et al., 2003]. L'une des justifications de cette approche repose sur l'observation que les heuristiques ont des performances qui varient en fonction de l'instance de problème traitée. Ainsi, l'approche Hyper-heuristique propose de regrouper un ensemble d'heuristiques ou de métaheuristiques et de mettre en place un mécanisme capable d'identifier et de sélectionner les méthodes de recherche les plus performantes au cours du processus d'optimisation.

Dans les hyper-heuristiques, les heuristiques sont des boîtes noires permettant de réaliser une recherche, et constituant la "couche basse" du système. L'hyper-heuristique est la "couche supérieure" qui sélectionne les heuristiques à appliquer en fonction d'informations non spécifiques au problème traité. Par exemple, le choix d'une heuristique peut être conditionné par le temps d'exécution moyen de cette dernière, ou encore, à la différence de fitness apportée en moyenne par l'heuristique lors de ses précédentes applications. Le mécanisme de sélection est donc indépendant du problème. Ce principe de séparation entre un

niveau hyper-heuristique indépendant du domaine du problème, et un niveau qui en dépend composé des heuristiques spécifiques, est illustré dans la figure 3.2.

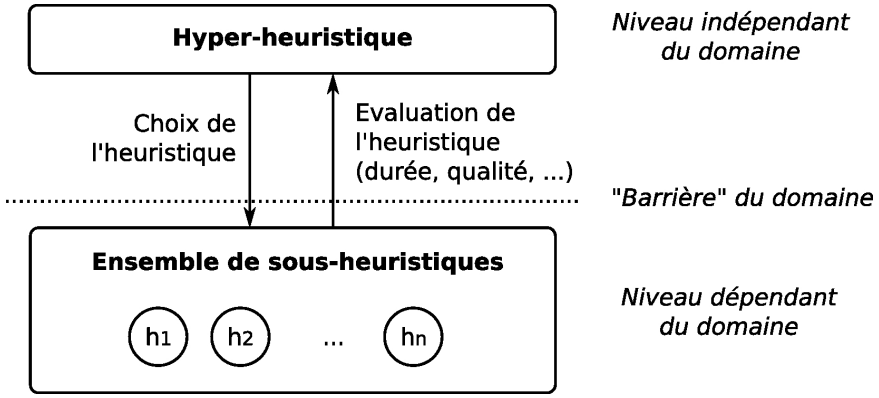

Figure 3.2: Schéma d'une hyper-heuristique, inspiré de [Burke et al., 2003]

Cowling et al. ont proposé dans [Cowling et al., 2001, 2000] une hyper-heuristique qui a été appliquée sur divers problèmes d'ordonnancement. Dans leur approche, le choix d'une heuristique est fondé sur trois critères notés $f_1(N_j)$, $f_2(N_j, N_k)$ et $f_3(N_j)$, où N_j et N_k sont des heuristiques. Le premier critère reflète les performances récentes de l'heuristique N_j en termes d'amélioration de la fitness rapportée au temps d'exécution. Le second critère $f_2(N_j, N_k)$, évalue les performances de l'application de l'heuristique N_j lorsqu'elle suit immédiatement l'application de N_k. Le troisième critère correspond à la durée écoulée depuis la dernière application de N_j. Le choix d'une heuristique N_j à un moment donné dépendra alors d'une combinaison de ces trois critères, exprimée généralement comme une somme pondérée. Cette approche, bien que posant des problèmes d'ajustement de l'intensification et de la diversification [Cowling et al., 2001, Özcan et al., 2006] via les pondérations choisies, semble apporter cependant une réponse simple et élégante à la problématique de l'adaptation dynamique de la stratégie de recherche.

3.5 Conclusion

Après avoir présenté quelques définitions consensuelles du concept d'agent et de système multi-agent, nous nous sommes attachés à décrire quelques applications de ce paradigme à la résolution de problèmes d'optimisation. Nous avons présenté tout d'abord des approches que l'on peut qualifier de méthodes heuristiques orientées-agents. Dans ce cas, il s'agit principalement d'heuristiques qui utilisent des métaphores spatiales et/ou naturelles, et dont le principe repose sur une décomposition du problème. Les agents représentent ou peuvent être associés à des variables ou contraintes, ou encore à des entités physiques telles que des véhicules, des clients ou des arrêts de bus.

Nous avons ensuite présenté des métaheuristiques qui sont également analysées en termes de recherche distribuée. Certaines approches fondées sur des métaphores d'insectes

sociaux sont mentionnées dans la littérature comme des exemples d'approches de résolution collective de problèmes et relèvent donc clairement du domaine des systèmes multi-agents. Cependant, des aspects d'autres métaheuristiques peuvent être interprétés du point de vue des systèmes multi-agents du fait de leur nature distribuée ou des mécanismes d'apprentissage utilisés.

À notre avis, l'adoption du paradigme multi-agent pour la conception de métaheuristiques repose essentiellement sur l'adoption d'une posture intentionnelle appliquée à des processus de recherche s'exécutant en parallèle et en interaction les uns avec les autres. C'est ce qui est sous-entendu lorsque l'on dit que le système est constitué d'agents autonomes possédant des buts et agissant collectivement en vue de la réalisation de ces buts. La plupart des approches métaheuristiques examinées incorporent à divers degrés cette vision.

Néanmoins, les métaphores de métaheuristiques sont variées et aucune d'elles ne semblent réunir simultanément les trois caractéristiques suivantes : posture intentionnelle, résolution distribuée mais aussi décentralisée du problème, et apprentissage collectif au sens de l'auto-adaptation des choix d'opérateurs. Les approches à base de colonies de fourmis supposent une mémoire partagée, les essaims particulaires n'incorporent pas l'apprentissage des choix d'opérateurs, les algorithmes mémétiques sont des algorithmes hybrides reposant sur une description séquentielle masquant un certain degré de centralisation du contrôle lors des sélections. Les autres métaheuristiques distribuées reposent seulement sur une mise en compétition de méthodes diverses. Enfin, les hyper-heuristiques n'incorporent pas l'aspect collectif de la résolution.

Toutes ces approches comportent néanmoins des points communs, qui ont été mis en avant dans les frameworks que nous avons présentés dans le chapitre précédent. En ce qui nous concerne, nous proposons d'approfondir l'application de la démarche orientée agent aux métaheuristiques en considérant cette perspective. Nous proposons de les analyser en adoptant certains outils méthodologiques généraux spécifiques à la conception de systèmes multi-agents. Ainsi, nous proposons dans le chapitre suivant un framework organisationnel, couvrant le domaine des métaheuristiques. Celles-ci seront présentées en termes d'organisations et de rôles en interaction joués par des agents. Puis, partant des points communs fondamentaux retenus, nous proposerons un modèle de métaheuristique multi-agent fondé sur une métaphore de coalition.

DEUXIÈME PARTIE

Conception orientée-agent de métaheuristiques

CHAPITRE 4

FRAMEWORK ORIENTÉ-AGENT POUR LA MODÉLISATION ET L'IMPLANTATION DE MÉTAHEURISTIQUES

Sommaire

4.1 Introduction

L'application d'une métaheuristique consiste généralement à sélectionner le schéma de métaheuristique proprement dit ou à en retenir parfois que certains des composants pour les assembler, puis les implanter en intégrant directement les caractéristiques du problème à traiter. Peu d'outils de conception permettent d'assister ce développement, et cette démarche, qui souvent repose sur l'expérience du concepteur, ne permet pas une bonne réutilisation, est sujette aux erreurs, et est finalement coûteuse en temps de développement [Birattari, 2005]. Afin de remédier à ces problèmes, il nous a semblé pertinent d'adopter une démarche unificatrice visant à réunir les aspects importants et essentiels des différentes métaheuristiques dans un cadre commun, celui des systèmes multi-agents. Le framework *AMF* (*Agent Metaheuristic Framework*) que nous proposons dans ce chapitre se situe dans cette perspective.

Nous utilisons le terme de "Framework" pour décrire un ensemble d'outils facilitant le développement d'applications. Un framework fournit un modèle ou un squelette assez générique pour être réutilisé lors de la conception de différentes applications. Dans [Roli and Milano, 2002], les auteurs proposent le framework *MAGMA* pour la conception de métaheuristiques. Ce framework particulier est présenté dans la section 2.5. D'après les auteurs, les principaux objectifs d'un framework de métaheuristiques sont :

- Comparer les algorithmes existants,
- Analyser les propriétés des algorithmes afin d'identifier leurs avantages et inconvénients,
- Faciliter la conception d'algorithmes hybrides ou de nouveaux algorithmes,
- Assister le développement d'applications.

Ainsi, en fournissant un cadre de modélisation commun, un framework doit faciliter l'analyse et la conception de métaheuristiques. Par exemple, les auteurs du framework *MAGMA* présentent un modèle où les composants de métaheuristiques se répartissent en 4 niveaux fonctionnels associés à des types d'agents spécifiques. Les auteurs illustrent l'utilité de leur framework en concevant un modèle de métaheuristique hybride original.

Le framework que nous proposons se distingue des frameworks existants, présentés dans la section 2.5, en introduisant les concepts permettant de traiter la distribution de la résolution et de l'(auto)-adaptation de la recherche. De plus, et en complément des approches existantes, un guide méthodologique détaillant le processus de développement est proposé. Pour atteindre ce but, nous adoptons une approche organisationnelle et multi-agent.

Un framework doit être conçu autour d'un ou plusieurs modèles. Celui que nous proposons dans *AMF* est un modèle organisationnel utilisant le méta-modèle RIO (Rôle Interaction Organisation, [Gruer et al., 2002]). Le méta-modèle RIO et le modèle organisationnel d'*AMF* sont présentés dans les deux sections suivantes de ce chapitre. Ensuite, dans la section 4.4 le guide méthodologique d'*AMF* est détaillé. Enfin, la dernière section donne une illustration du framework *AMF* au travers de différentes métaheuristiques existantes.

4.2 Approche organisationnelle et méta-modèle RIO

La conception d'un framework exige l'élaboration d'un ou plusieurs modèles. Afin, d'adopter une approche multi-agent dans *AMF*, nous avons conçu un modèle organisationnel de métaheuristiques. L'intérêt de l'approche organisationnelle est de pouvoir décrire un système aussi bien comme un tout, le système multi-agents, que comme l'assemblage de composants, les agents. De plus, cette approche permet de distinguer l'analyse des fonctions du système de l'analyse de l'architecture. Enfin, l'approche organisationnelle encourage la modularité et la réutilisation des modèles. Pour concevoir le modèle organisationnel de métaheuristiques, le méta-modèle RIO [Gruer et al., 2002] a été utilisé.

RIO tient son nom des trois concepts fondamentaux à l'origine du méta-modèle : Rôle, Interaction et Organisation. Dans le cadre de RIO, une organisation est une structure regroupant un ensemble de rôles en interaction. Chaque organisation est associée à un objectif à satisfaire, une tâche à effectuer, ou un comportement global à exhiber. L'objectif d'un rôle est de contribuer pour tout ou partie aux besoins associés à l'organisation dans laquelle il est défini. La définition du concept de rôle donnée dans [Hilaire, 2000] est la suivante :

Définition 4.1 Rôle, [Hilaire, 2000]

Un rôle est l'abstraction d'un comportement ou un modèle de conduite, rattaché à un statut, qui peut interagir avec d'autres rôles.

Il ressort de cette définition les notions de comportement et de statut. Le statut d'un rôle définit la position de ce dernier au sein de son organisation ainsi qu'un ensemble de droits et d'obligations pour l'agent qui le joue. Le comportement d'un rôle détermine comment les objectifs du rôle peuvent être satisfaits et détaille comment sont combinées et ordonnées les interactions avec les autres rôles. Cependant, un rôle est indépendant de l'entité qui le met en oeuvre. Une interaction entre rôles est définie de la manière suivante :

Définition 4.2 Interaction, [Gaud, 2007]

Une interaction entre rôles est composée de l'événement généré par un premier rôle et perçu par les autres, ainsi que les réactions induites dans ces derniers. Les rôles qui interagissent partagent automatiquement un contexte d'interaction commun.

Le rôle et l'interaction sont les concepts fondamentaux définissant une organisation. Ainsi, une organisation peut être définie de la manière suivante.

Définition 4.3 Organisation, inspiré de [Hilaire, 2000]

Une organisation est définie par un ensemble de rôles, leurs interactions et un contexte commun, le tout formant un point de vue particulier d'une fonction globale pour un problème donné.

À partir des concepts de rôle, d'interaction et d'organisation, un agent est simplement défini comme une entité active et communicante jouant des rôles. Un rôle peut être affecté à plusieurs agents et un agent peut jouer plusieurs rôles. Les agents associés aux rôles d'une organisation font partie d'un groupe. Un groupe est une instance concrète d'une organisation où un ensemble d'agents interagissent afin de satisfaire les objectifs de l'organisation.

La figure 4.1 illustre les deux niveaux essentiels de la conception de système multi-agents par une approche organisationnelle. Le niveau organisationnel présente l'organisation, les rôles et les interactions, quant au niveau agent, il décrit l'architecture du système multi-agent et l'affectation des rôles entre les agents. L'exemple considéré dans la figure 4.1 est la modélisation d'un système destiné à la gestion de projets (*Project Management*) tel qu'il est présenté dans [Gaud, 2007].

Dans l'organisation *Project Management*, deux rôles sont identifiés : *Manager* et *Participant*. Ces rôles illustrent les notions de statut et de comportement. En effet, le rôle *Manager* décrit des obligations envers le rôle *Participant* et permet de détailler le comportement associé à la tâche de management. Le rôle *Manager* autorise l'agent qui le joue à assigner les tâches aux agents jouant le rôle *Participant*. Ce droit correspond au statut associé au rôle *Manager*. Le comportement de ce rôle décrit la manière d'effectuer la répartition des tâches parmi les agents jouant le rôle *Participant*.

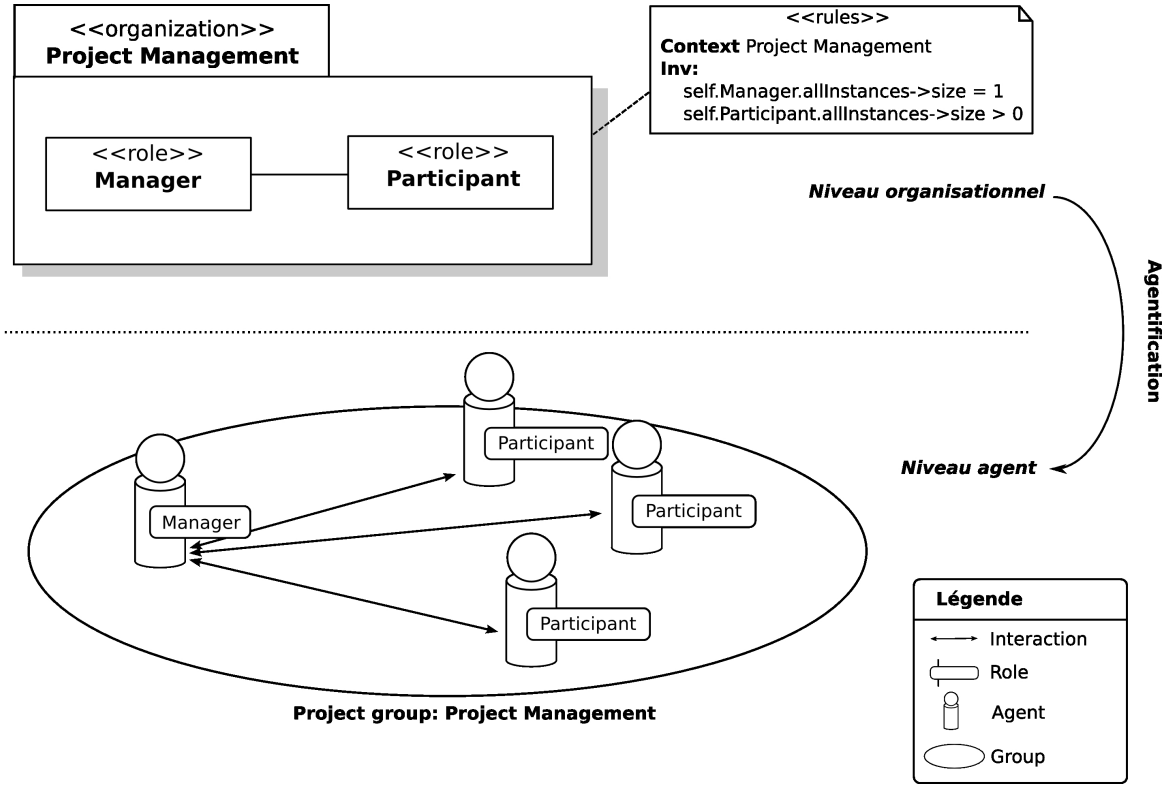

Figure 4.1: Diagramme de l'organisation *Gestion de projet* et de son instance *Groupe de projet*

Il est possible d'affecter certaines contraintes à l'organisation. Nous décrivons ces contraintes

sous la forme de règles *OCL*[1]. Ici, les règles associées à l'organisation *Project Management* précisent que les groupes issus de l'organisation *Project Management* contiennent une seule instance du rôle *Manager* et au moins une instance du rôle *Participant*.

Le modèle de cette organisation décrit dans la partie supérieure de la figure 4.1 n'établit pas encore la structure du système multi-agent. Celle-ci peut être décrite a posteriori en associant les rôles à des agents. Le niveau agent de la figure 4.1 précise la manière dont sont répartis les rôles aux agents. Le passage entre le niveau organisationnel et le niveau agent est nommé agentification. Il s'agit de l'instanciation de l'organisation sous la forme d'un ou de plusieurs groupes. Dans l'exemple de la figure 4.1, un groupe est défini. Ce groupe contient trois agents jouant le rôle *Participant* et un agent qui joue le rôle *Manager*. L'agent *Manager* interagit avec les agents *Participant* suivant l'interaction qui a été définie au niveau organisationnel.

4.3 Modèle organisationnel de métaheuristiques

À l'aide du méta-modèle RIO, nous avons introduit les concepts permettant de modéliser de manière organisationnelle un système. À partir d'un modèle organisationnel, il est ensuite possible de décrire un système multi-agent où chaque agent est associé à un ensemble de rôles. Dans cette section, nous nous appuyons sur l'approche RIO pour proposer un modèle organisationnel des métaheuristiques.

4.3.1 Rôles et interactions au sein d'une métaheuristique

Afin d'obtenir un modèle de métaheuristique, nous considérons cette dernière comme une organisation (au sens du méta-modèle RIO) dont l'objectif est d'explorer l'espace de recherche d'une instance de problème afin de trouver une solution optimale ou proche de l'optimale. Au sein de cette organisation, il est possible d'identifier des tendances ou des mécanismes distincts d'intensification et de diversification. L'intensification permet de concentrer la recherche dans les zones prometteuses et la diversification sert à découvrir des zones de l'espace de recherche non encore explorées. Ces deux tendances sont guidées par un ensemble d'informations structurées, portant sur l'espace de recherche. Un mécanisme supplémentaire gérant ces informations permet de coordonner et d'équilibrer l'intensification et la diversification. De plus, pour certaines métaheuristiques, la stratégie de recherche peut être adaptée dynamiquement en fonction du contexte d'optimisation et des expériences de recherche.

À partir de cette description succincte des métaheuristiques, il est possible d'identifier dans l'organisation "Métaheuristique" quatre rôles fondamentaux que nous nommerons :

[1]OCL : *Object Constraint Language*, `http://www.omg.org/technology/documents/formal/ocl.htm`

Intensifieur, Diversifieur, Guide et Stratège. Les rôles Intensifieur et Diversifieur sont relatifs aux tendances ou mécanismes d'intensification et de diversification. Le rôle Guide décrit la coordination ainsi que la recherche d'équilibre entre l'intensification et la diversification, et le rôle Stratège correspond à l'adaptation. L'organisation issue de cette description est illustrée par la figure 4.2.

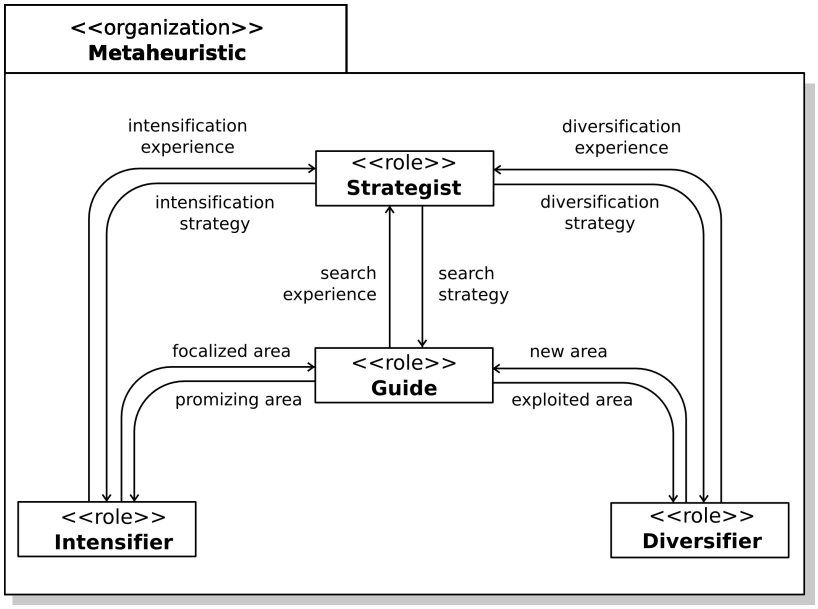

Figure 4.2: Diagramme du modèle organisationnel de métaheuristique

Les quatre rôles décrits dans ce modèle organisationnel peuvent être facilement identifiés dans plusieurs métaheuristiques. Par exemple, la recherche locale itérée est une métaheuristique qui s'articule autour de trois procédures : une recherche locale qui permet d'obtenir un minimum local, une procédure de perturbation pour sortir de ce minimum, et un critère d'acceptation pour mettre à jour la meilleure solution trouvée. Ces trois procédures peuvent être associées respectivement aux rôles Intensifieur, Diversifieur et Guide. Dans le cadre de la recherche locale itérée de base, le rôle Stratège n'est pas présent. Cependant, certains mécanismes d'adaptation ou d'auto-adaptation tels que l'ajustement de l'intensité de la perturbation décrit dans [Lourenço et al., 2003], peuvent être associés à ce dernier rôle. Nous détaillons chaque rôle du modèle organisationnel proposé dans les sections suivantes.

Le modèle de métaheuristique proposé ne décrit pas de manière formelle l'ensemble des métaheuristiques. Il s'agit plutôt d'un schéma de modélisation à la manière des "design patterns" utilisés en modélisation orientée-objet [Gamma et al., 1999]. Ce modèle doit être raffiné en fonction des caractéristiques particulières du problème à traiter. L'intérêt du modèle organisationnel d'*AMF* est d'introduire un ensemble de concepts assez généraux pour être communs aux différentes métaheuristiques et prenant la forme de rôles ou d'interactions. Cette généralisation de concepts doit permettre de faciliter l'analyse et la conception de métaheuristiques [Meignan et al., 2008b].

Pour faciliter la représentation de modèles organisationnels de métaheuristiques issus du raffinement du modèle précédent, nous introduisons la notion de "stéréotypes" utilisée dans les diagrammes de classes *UML*. Ces stéréotypes notés entre "<<" et ">>" permettent d'identifier les différents rôles issus du raffinement du modèle organisationnel d'*AMF*. Par exemple, un raffinement du rôle "Guide" sera identifié dans nos diagrammes par l'étiquette *<<Guide role>>*. Ainsi, dans AMF nous utilisons les stéréotypes *<<role>>* et *<<organization>>* introduits dans RIO [Gaud, 2007] et nous définissons cinq stéréotypes spécifiques à la modélisation organisationnelle de métaheuristiques. Le stéréotype *<<Metaheuristic model>>* s'applique à un *package* et identifie un modèle organisationnel de métaheuristique. Les stéréotypes *<<Intensifier role>>*, *<<Diversifier role>>*, *<<Guide role>>* et *<<Strategist role>>* s'appliquent aux classes respectivement associées aux rôles Intensifieur, Diversifieur, Guide et Stratège.

Avant de détailler les étapes permettant de concevoir une métaheuristique à partir du modèle proposé, nous définissons de manière plus détaillée les différents rôles que nous avons identifiés.

4.3.2 Les rôles Intensifieur et Diversifieur

L'intensification et la diversification sont les objectifs respectifs des rôles Intensifieur et Diversifieur. Les termes "exploitation" et "exploration" peuvent aussi être employés pour faire référence aux objectifs de ces rôles. Dans [Blum and Roli, 2003], les auteurs font une revue des différentes définitions données aux concepts d'intensification et de diversification. Il ressort de ces définitions que l'intensification favorise l'exploration des zones de l'espace de recherche où des solutions de bonne qualité ont été trouvées. La diversification permet de déplacer la recherche dans les zones non explorées de l'espace de recherche.

L'intensification et la diversification sont souvent vues comme deux tendances contradictoires. Ainsi, Battiti [Battiti, 1996] décrit l'intensification et la diversification comme étant deux besoins conflictuels dont l'équilibre est essentiel au bon fonctionnement d'une heuristique. Par exemple, dans les métaheuristiques de trajectoire une trop forte intensification bloque la recherche dans les minimums locaux et une diversification trop importante engendre une recherche trop longue.

Ces deux tendances sont parfois simples à identifier lorsqu'il s'agit de mécanismes distincts comme dans la recherche locale itérée. Cependant, il est plus difficile de discerner l'intensification et la diversification dans certaines métaheuristiques telles que la recherche tabou ou les algorithmes de colonies de fourmis. En effet, dans les versions de base de ces deux méthodes, l'intensification et la diversification ne correspondent pas à des phases de recherche ou des procédures distinctes. Les deux tendances sont combinées dans un même composant ou mécanisme.

Pour distinguer l'intensification et la diversification, nous nous rapportons à la définition donnée dans le cadre du framework *I&D Frame* [Blum and Roli, 2003]. Dans ce framework,

les métaheuristiques sont analysées au travers de composants d'intensification/diversification (*I&D Components*). Les auteurs observent que l'intensification est une recherche guidée par la fonction objectif alors que la diversification est fondée sur l'aléatoire ou une fonction autre que la fonction objectif. Par exemple, dans les algorithmes de colonies de fourmis, lorsqu'une solution est générée par une fourmi, la composante heuristique ou aléatoire du choix des arrêtes dans le graphe est liée à la diversification alors que la composante entraînant le choix vers l'arrête à plus fort taux de phéromone correspond à l'intensification. En effet, le taux de phéromone est lié aux précédentes évaluations des solutions.

À partir de ces observations, nous donnons les définitions suivantes aux rôles Diversifieur et Intensifieur.

Définition 4.4 Rôle Intensifieur

Le rôle Intensifieur a pour objectif l'intensification de la recherche. L'intensification permet de concentrer la recherche dans les zones prometteuses (de meilleure qualité) de l'espace de recherche. Le rôle Intensifieur exploite pour cela des informations sur les zones précédemment explorées et utilise la fonction objectif du problème pour guider la recherche.

Définition 4.5 Rôle Diversifieur

Le rôle Diversifieur a pour objectif la diversification de la recherche. La diversification permet de déplacer la recherche dans les zones non explorées de l'espace de recherche. La diversification de la recherche est fondée sur l'aléatoire ou une fonction autre que la fonction objectif. Le rôle Diversifieur exploite éventuellement les informations sur les zones précédemment explorées pour s'en éloigner.

Dans ces définitions, les informations correspondant aux zones de l'espace de recherche (zones prometteuses, zones précédemment explorées, zones non explorées) peuvent prendre différentes formes. Donnons en deux exemples en considérant la recherche locale itérée et l'optimisation par colonie de fourmis. Dans la recherche locale itérée, le point de départ de l'intensification est une solution perturbée. Celle-ci correspond à une "zone prometteuse". Par contre, dans l'optimisation de colonies de fourmis, les "zones prometteuses" correspondent aux traces de phéromone. Si la nature de ces informations diffère, leur objectif est identique. Ainsi, nous avons introduit cette notion de "zone" par souci de généricité.

4.3.3 Le rôle Guide

Le rôle Guide met en oeuvre une stratégie globale de recherche en s'appuyant sur les rôles Intensifieur et Diversifieur précédemment définis. Les objectifs du rôle Guide sont de coordonner les rôles Intensifieur et Diversifieur, d'équilibrer le rapport entre les tendances

d'intensification et de diversification, et de sélectionner la ou les solutions qui seront le résultat de l'optimisation.

Ce rôle peut être assimilé au niveau 2 du modèle *MAGMA* [Milano and Roli, 2004] présenté dans la section 2.5. Comme nous l'avons déjà souligné, *MAGMA* présente les métaheuristiques en considérant quatre niveaux. Les niveaux 0 et 1 sont chargés de construire et d'améliorer les solutions, et le niveau 2 se charge de contrôler le rapport diversification/intensification en coordonnant les niveaux inférieurs. Le niveau 3 permet de coordonner plusieurs processus de recherche. Le niveau 2 correspond donc au rôle Guide dans notre modèle.

Pour effectuer la coordination entre intensification et diversification, le rôle Guide utilise une mémoire. Nous utilisons le terme de "mémoire" en référence à l'approche *AMP* [Taillard et al., 2001]. Dans cette approche, la mémoire permet de guider le processus de recherche et constitue un élément central des métaheuristiques. Elle prend différentes formes selon la métaheuristique considérée. Par exemple, dans la recherche tabou, il s'agit de la liste des solutions tabou. Dans l'optimisation par colonie de fourmis, elle correspond à la matrice de phéromone. Les auteurs de l'approche *AMP* distinguent deux procédures s'appliquant à la mémoire : la génération d'une solution temporaire et la mise à jour à partir d'une nouvelle solution. Nous généralisons ces "interactions" avec la mémoire en la définissant dans *AMF* comme un composant permettant :

- de fournir au rôle Intensifieur les informations sur les zones prometteuses,
- de fournir éventuellement au rôle Diversifieur les informations concernant les zones de l'espace de recherche déjà explorées,
- de combiner les résultats de l'intensification et de la diversification.

Ces différentes interactions entre les rôles Guide, Intensifieur et Diversifieur, sont reportées sur la figure 4.2.

En reprenant l'exemple de la recherche locale itérée, les procédures de recherche locale et de perturbation sont associées respectivement aux rôles Intensifieur et Diversifieur. Le rôle Guide consiste à alterner l'intensification et la diversification et choisir la meilleure solution suivant un critère d'acceptation. La mémoire relative au rôle Guide est composée de la meilleure solution trouvée. Lorsque le rôle Guide interagi avec le rôle Intensifieur, la solution obtenue par la recherche locale permet de mettre à jour cette meilleure solution trouvée. Pour les interactions entre le rôle Guide et Diversifieur, les informations sur les zones explorées correspondent à la meilleure solution trouvée.

Pour résumer ces différents éléments, nous donnons la définition suivante au rôle Guide :

Définition 4.6 Rôle Guide

Le rôle Guide met en oeuvre une stratégie globale de recherche en coordonnant les rôles Intensifieur et Diversifieur. Ce rôle est chargé d'équilibrer les tendances d'intensification et de diversification. De plus, le rôle Guide sélectionne la ou les solutions qui seront le résultat du processus d'optimisation. Pour effectuer ces différentes tâches, le rôle Guide gère une mémoire à partir de laquelle il est possible d'extraire des informations sur les zones prometteuses de l'espace de recherche et éventuellement sur les zones explorées. Cette mémoire est mise à jour en combinant les résultats de l'intensification et de la diversification.

4.3.4 Le rôle Stratège

Il existe plusieurs versions améliorées de métaheuristiques auxquelles il a été ajouté des mécanismes d'adaptation de la stratégie de recherche. Ces métaheuristiques sont souvent qualifiées d'adaptatives ou auto-adaptatives. Par exemple, dans l'approche *RTS* (*Reactive Tabu Search*) [Battiti and Tecchiolli, 1994], une extension de la recherche tabou, la taille de la liste tabou est modifiée dynamiquement. Dans l'approche *SAGA* (*Self-Adaptive Genetic Algorithm*) [Hinterding et al., 1996] fondée sur les algorithmes génétiques, la taille de la population ainsi que les taux de mutation sont modifiés dynamiquement. Ces mécanismes d'adaptation permettent d'ajuster la stratégie de recherche en fonction des expériences de recherche. Dans le modèle organisationnel d'*AMF*, ces mécanismes sont pris en compte par le rôle Stratège.

Afin de décrire le rôle Stratège, il est nécessaire de distinguer la notion d'adaptation au sens large de l'adaptation de la stratégie de recherche.

En général, l'adaptation est la capacité d'un système ou d'un organisme à évoluer afin d'acquérir un comportement approprié à son environnement. Cette définition s'applique aussi bien aux systèmes naturels qu'artificiels. Pour les métaheuristiques, l'adaptation est une caractéristique essentielle. L'exemple le plus évident, de l'utilisation de la notion d'adaptation dans les métaheuristiques est sûrement les algorithmes évolutionnistes où l'on parle d'adaptation de la population au fil des générations. Cependant, cette définition est trop large pour caractériser les processus particuliers d'adaptation intervenant dans l'approche *RTS* ou l'approche *SAGA*. En effet, un algorithme de recherche tabou et un algorithme d'optimisation par colonie de fourmis simples peuvent être qualifiés d'adaptatifs, dans le sens où ils utilisent une mémoire qui rassemble des connaissances acquises au cours de la recherche, et qui guide le processus de recherche [Devarenne, 2007]. Nous donnons donc une définition particulière à la notion d'adaptation de la stratégie de recherche qui caractérise des processus tels que la modification dynamique de la taille de la liste tabou dans l'approche *RTS* ou la modification de la taille de la colonie de fourmis dans l'approche *SAGA*. Cette définition de l'adaptation dans les métaheuristiques permettra de définir les objectifs du rôle Stratège.

Nous définissons l'adaptation dans *AMF* comme la modification ou l'ajustement du processus de recherche. L'adaptation est caractérisée par le fait que l'élément affecté par la modification ou l'intensité de la modification n'est pas connu à l'avance. La nature de la modification est issue d'un feed-back (rétroaction) de la recherche. Ainsi, le processus d'adaptation peut se traduire par :

l'observation des expériences de recherche : les données prisent en considération par le processus d'adaptation ne sont pas uniquement les résultats de la recherche, mais aussi les actions qui ont permis d'obtenir ces résultats.

l'évaluation de ces expériences : les expériences observées sont évaluées de manière explicite ou implicite afin d'identifier si les actions mises en place par la stratégie de recherche sont efficaces ou non.

la modification de la stratégie de recherche : l'adaptation entraîne une modification de la stratégie de recherche. Cette modification est issue de l'évaluation des expériences.

L'auto-adaptation est un cas particulier de l'adaptation. On parlera d'auto-adaptation lorsque le processus d'adaptation est combiné au processus de recherche [Hinterding et al., 1997]. Par exemple, dans les algorithmes évolutionnistes, il est possible de faire évoluer les paramètres du processus d'optimisation en considérant ces derniers comme des éléments associés aux individus. L'évolution de ces paramètres est combinée à l'évolution des solutions, on parlera alors d'auto-adaptation. D'autres approches adaptatives des algorithmes évolutionnistes consistent à mettre en concurrence plusieurs sous-populations avec différents paramétrages. Dans ce cas, le mécanisme qui modifie les paramètres est distinct du processus de recherche, et on qualifiera ce type d'approche d'adaptative.

Dans le modèle organisationnel d'*AMF*, le rôle Stratège a pour objectif d'adapter la stratégie de recherche. Les interactions de ce rôle avec les rôles Intensifieur, Diversifieur et Guide consistent, à observer les expériences de recherche, et en retour, à modifier la stratégie de recherche. Dans la plupart des cas, l'adaptation de la stratégie de recherche correspond à l'ajustement de paramètres stratégiques comme nous l'avons évoqué plus haut avec les approches RTS et SAGA.

À partir de cette définition de l'adaptation, le rôle Stratège est défini de la manière suivante :

Définition 4.7 Rôle Stratège

Le rôle Stratège a pour objectif d'adapter la stratégie de recherche. Pour cela, il interagit avec les rôles Intensifieur, Diversifieur et Guide pour observer leurs expériences et ajuste en conséquence les comportements.

4.4 Guide méthodologique pour la modélisation de métaheuristiques

Dans la section précédente a été présenté un modèle organisationnel de métaheuristiques. Ce modèle décrit des concepts communs aux différentes métaheuristiques. L'objectif est de faciliter la conception de métaheuristiques en adoptant une approche organisationnelle et multi-agent. Dans cette section, nous proposons quelques éléments méthodologiques pour la conception de métaheuristiques en nous appuyant sur le modèle organisationnel d'*AMF*. Pour cela, nous nous inspirons du processus de conception associé à RIO [Gruer et al., 2002].

Les auteurs du méta-modèle RIO proposent un processus de conception associé au méta-modèle RIO. Ce processus est composé de trois phases : spécification, agentification et conception de l'architecture des agents.

Dans la phase de spécification, le système est vu sous la forme d'une organisation regroupant un ensemble de rôles en interaction. Cette phase consiste à décrire l'ensemble des rôles et des interactions en décomposant éventuellement les rôles complexes en plusieurs autres rôles en interaction. Une fois que l'organisation est complètement spécifiée, l'agentification consiste à définir les agents composant le système et à leur assigner des rôles. Étant donné que les agents peuvent jouer plusieurs rôles de manière dynamique, l'agentification doit préciser la manière dont les agents ordonnancent les comportements des différents rôles. Enfin, la phase de conception de l'architecture des agents a pour but de décrire l'architecture interne des agents.

Les règles méthodologiques associées au modèle organisationnel d'*AMF* s'inspirent du processus de conception RIO. Elles permettent, à partir du modèle organisationnel d'*AMF*, d'obtenir un système multi-agent instanciant une métaheuristique. Ce processus de conception se compose lui aussi de trois phases :

- raffinement du modèle organisationnel d'*AMF*
- agentification du modèle organisationnel de métaheuristique
- spécialisation de la métaheuristique

Contrairement au processus de conception RIO, le processus proposé part d'un modèle organisationnel existant. De plus, nous intégrons la notion de spécialisation que nous définissons comme étant le passage d'une méthode d'optimisation décrite à un niveau indépendant du problème à une méthode conçue pour résoudre un problème d'optimisation donné. Dans la suite, nous explicitons ces trois phases.

Raffinement du modèle organisationnel d'*AMF* : Cette phase de raffinement consiste à élaborer un modèle organisationnel de métaheuristique à partir du modèle organisationnel d'*AMF*. Pour obtenir le modèle issu de cette phase, il est nécessaire de choisir la méthode d'optimisation qui sera utilisée ou les différents composants de métaheuristiques qui seront hybridés. Le résultat de cette phase est la description d'une or-

ganisation de métaheuristique particulière où l'ensemble des rôles et des interactions est décrit.

Agentification du modèle organisationnel de métaheuristique : La seconde étape a pour objectif de décrire la structure du système multi-agent associé à la métaheuristique. Pour cela il est nécessaire d'identifier les différents types d'agents composant le système multi-agent, de préciser l'affectation des rôles aux agents et de décrire l'ordonnancement des rôles pour chaque type d'agent. La principale donnée d'entrée pour effectuer cette étape correspond au modèle raffiné de la métaheuristique précédemment obtenue.

Spécialisation de la métaheuristique : La troisième étape consiste à spécialiser les éléments génériques de la métaheuristique en fonction du problème d'optimisation à traiter. Par exemple, si un algorithme génétique a été décrit sous la forme d'un système multi-agent, la spécialisation correspond à la définition des individus et des opérateurs de mutation et de croisement spécifiques au problème d'optimisation traité.

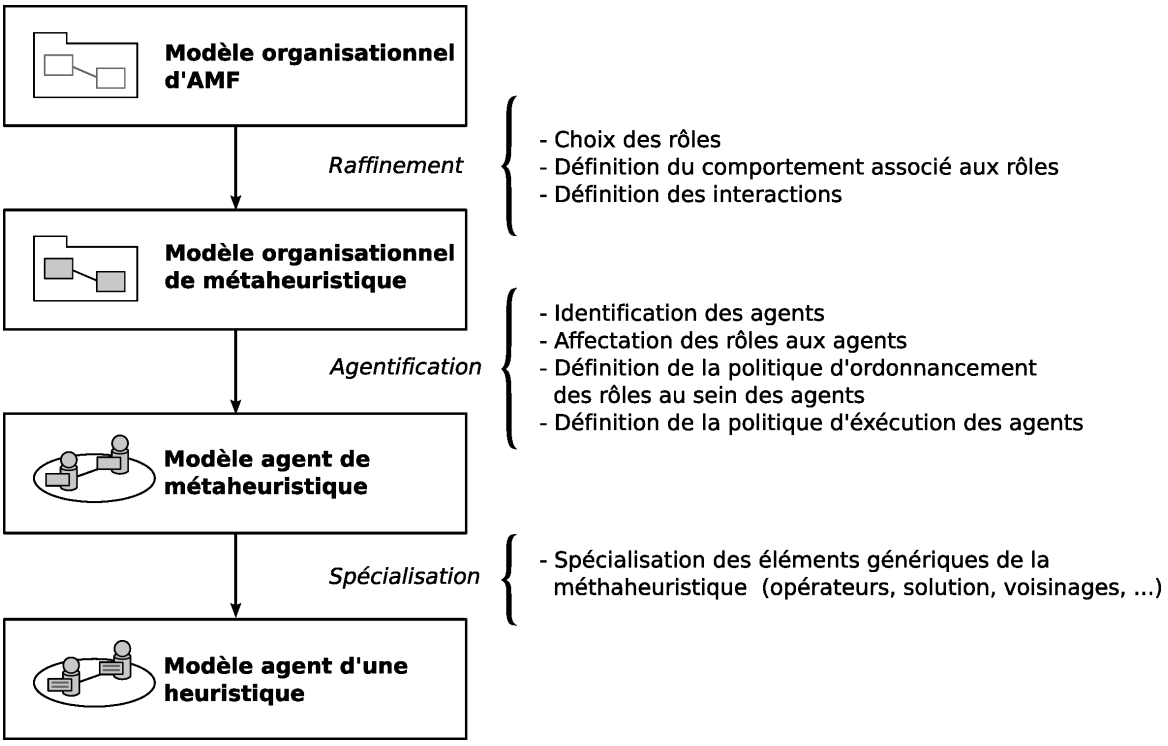

Figure 4.3: Phases de modélisation d'une métaheuristique

Ce processus de modélisation est résumé dans la figure 4.3. À l'issu de ce processus est obtenu un modèle de système multi-agent pouvant être implémenté en utilisant éventuellement une plateforme multi-agent.

Le processus de modélisation proposé favorise la réutilisation à différents niveaux. En effet, lors de la phase de raffinement, il est possible de réutiliser des rôles issus de métaheuristiques existantes. De plus, la distinction d'une phase de spécialisation permet de réutiliser un système multi-agent pour traiter différents problèmes d'optimisation.

4.5 Illustration du modèle organisationnel d'*AMF*

Dans cette section, nous montrons l'utilisation du modèle organisationnel d'*AMF* en considérant trois métaheuristiques : la recherche locale itérée, les algorithmes évolutionnistes et l'optimisation par colonies de fourmis. Ces trois méthodes ont été choisies pour illustrer *AMF* car elles paraissent assez différentes. La recherche locale itérée est une méthode de trajectoire alors que les algorithmes évolutionnistes et l'optimisation par colonies de fourmis sont des méthodes à base de population. De plus, l'optimisation par colonies de fourmis, contrairement aux deux autres méthodes, est une métaheuristique fondée sur la construction de solutions.

Pour présenter ces trois métaheuristiques dans le cadre du modèle organisationnel, nous allons définir pour chacune d'elles une organisation dans laquelle les rôles correspondent à une réalisation concrète des rôles Intensifieur, Diversifieur, Guide et Stratège. Ce passage entre le "modèle conceptuel" (ou *pattern*) de l'organisation et le modèle concret correspond au raffinement qui est la première phase des règles méthodologiques d'*AMF*. De plus, à partir des modèles raffinés obtenus, nous proposons une agentification possible de chacune des métaheuristiques. Cette agentification illustre la seconde phase décrite dans la section précédente.

À la suite de ces trois exemples approfondis, le modèle organisationnel d'*AMF* est utilisé pour décrire succinctement les métaheuristiques les plus répandues. Cette analyse permet de montrer comment le modèle proposé unifie les différents concepts communs aux métaheuristiques et facilite l'analyse de ces dernières.

4.5.1 Recherche locale itérée

La recherche locale itérée est une métaheuristique de trajectoire. Son principe consiste à alterner une recherche locale afin d'obtenir un minimum local, et une procédure de perturbation afin de s'échapper de ce minimum. Lors de la recherche, la meilleure solution est mémorisée et sert de point de départ pour la procédure de perturbation.

Dans la partie supérieure de la figure 4.4 un modèle organisationnel de la recherche locale itérée est donné. Ce modèle correspond à un raffinement du modèle organisationnel d'*AMF*. Les rôles identifiés sont les suivants :

Rôle *Local Searcher* : Dans la recherche locale itérée, l'intensification est effectuée par recherche locale. Ainsi, le rôle *Local Searcher* correspond à un raffinement du rôle

Intensifier et a pour objectif de trouver un minimum local à partir d'une solution perturbée. Ce rôle ne peut avoir qu'une seule instance dans les groupes qui instancient l'organisation *Iterated Local Search*. Cette contrainte est spécifiée par les règles *OCL* associées à l'organisation.

Rôle *Perturbator* : Le rôle *Perturbator* est un raffinement du rôle *Diversifier*. Son objectif est de diversifier la recherche en appliquent une procédure de perturbation à un minimum local.

Rôle *Coordinator* : Le rôle *Coordinator* est chargé de coordonner l'intensification et la diversification. Ce rôle correspond à un raffinement du rôle *Guide*. Son comportement consiste à soumettre alternativement une solution au rôle *Local Searcher* puis au rôle *Perturbator*. Après l'obtention d'un minimum local, le rôle *Coordinator* décide de garder la solution ou de repartir du minimum local mémorisé en fonction d'un critère d'acceptation. La mémoire relative au rôle Guide est donc ici composée d'une unique solution.

Dans la version de base de la recherche locale itérée présentée dans la figure 4.4 le rôle Stratège n'est pas présent. Il existe peu d'extensions adaptatives de la recherche locale itérée et donc peu d'illustrations du rôle Stratège. Ceci peut être expliqué par une volonté de garder la simplicité de la méthode. De plus, les extensions relèvent plus de la recherche tabou ou des algorithmes mémétiques. Cependant, dans [Lourenço et al., 2003] et [Battiti and Protasi, 1996], les auteurs proposent une extension adaptative de la recherche locale itérée où l'intensité de la perturbation est ajustée en fonction des expériences de recherche. Dans ce cas précis, le rôle Stratège observe les expériences de recherche et interagit avec le rôle *Perturbator* afin de modifier le paramètre stratégique relatif à l'intensité de la perturbation.

Il n'y a que peu d'intérêt de distribuer une méthode de recherche locale itérée simple. C'est pour cela que nous présentons dans la partie inférieure de la figure 4.4 une version non distribuée de cette métaheuristique. Dans cette alternative, l'agentification consiste à affecter l'ensemble des rôles à un unique agent. L'intérêt de ce premier exemple est essentiellement porté sur la définition d'un modèle organisationnel de la recherche locale itérée.

4.5.2 Algorithmes évolutionnistes

Un algorithme évolutionniste est une métaheuristique utilisant la métaphore de l'évolution des espèces et de la sélection naturelle. Le principe est de faire évoluer une population d'individus (solutions) en appliquant des procédures de croisement, de mutation et de sélection. Chaque nouvelle génération d'individus est obtenue en croisant et mutant les individus de la génération courante, puis en appliquant une sélection sur la population composée des nouveaux et des anciens individus.

Dans les algorithmes évolutionnistes, l'intensification et la diversification peuvent être décrites selon plusieurs points de vues. Eiben et Schippers les détaillent dans [Eiben and

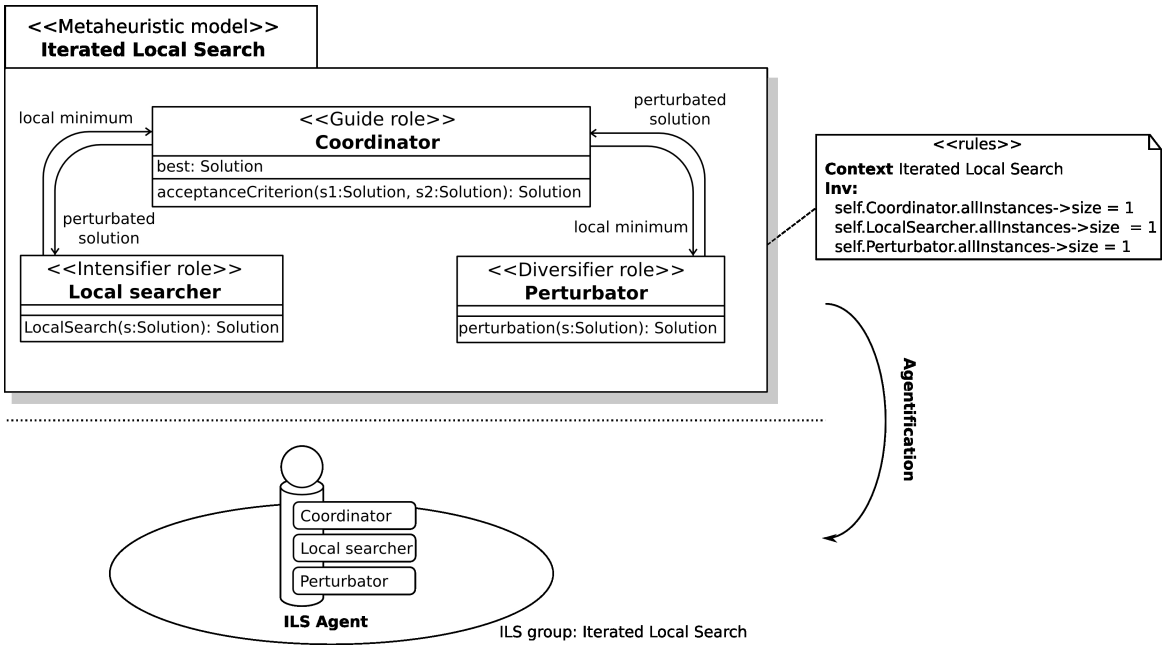

Figure 4.4: Un modèle organisationnel de la méthode de recherche locale itérée et son agentification

Schippers, 1998]. Il ressort de cette étude que les opérateurs de mutation et de croisement permettent d'explorer l'espace de recherche. Ces opérateurs agissent de manière aléatoire sur les individus sans nécessiter l'utilisation de la fonction objectif. Ainsi, la diversification peut être associée à ces deux types d'opérateurs. L'intensification est effectuée par la sélection des individus. Cette sélection est fondée sur la fonction de fitness des individus (évaluation des solutions) et consiste à favoriser la conservation des meilleurs individus. Ce principe d'intensification par la sélection de solutions est connu sous le nom de pression sélective. Certaines études ne partagent pas ce point de vue où l'opérateur de croisement est uniquement associé à l'exploration de l'espace de recherche. L'objectif de cet opérateur est parfois énoncé comme étant la combinaison des meilleures composantes des individus. À ce titre, le croisement peut être considéré comme étant un opérateur d'intensification. Ce point de vue est particulièrement soutenu dans le cadre des approches à base de schèmes ou building-blocks dans les algorithmes génétiques [Eshelman, 1997, Goldberg, 1994].

Dans la partie supérieure de la figure 4.5, un modèle organisationnel d'algorithme évolutionniste est représenté. Il est composé de trois rôles : *Recombinator - Mutator*, *Selector* et *Coordinator*. Ce modèle adopte le point de vue énoncé dans [Eiben and Schippers, 1998] pour distinguer l'intensification et la diversification. La définition des trois rôles est la suivante.

Rôle *Recombinator - Mutator* : Le rôle *Recombinator - Mutator* est un raffinement du rôle *Diversifier*. Il consiste à diversifier une population de solutions en appliquant des opérateurs de croisement et de mutation.

Rôle *Selector* : L'intensification de la recherche est obtenue par la sélection. Ainsi, le rôle

Selector, raffinement du rôle *Intensifier*, est chargé de sélectionner les individus d'une population en favorisant la conservation des meilleurs individus.

Rôle *Coordinator* : Dans les algorithmes évolutionnistes, la mémoire est constituée d'une population d'individus. Ces individus évoluent grâce aux opérateurs de mutation et de croisement et sont mis en compétition dans le cadre de la sélection. Le comportement du rôle *Coordinator*, raffinement du rôle *Guide*, consiste à soumettre la population aux croisements et mutations en interagissant avec le rôle *Recombinator - Mutator*, puis à obtenir la nouvelle génération d'individus par l'intermédiaire du rôle *Selector*.

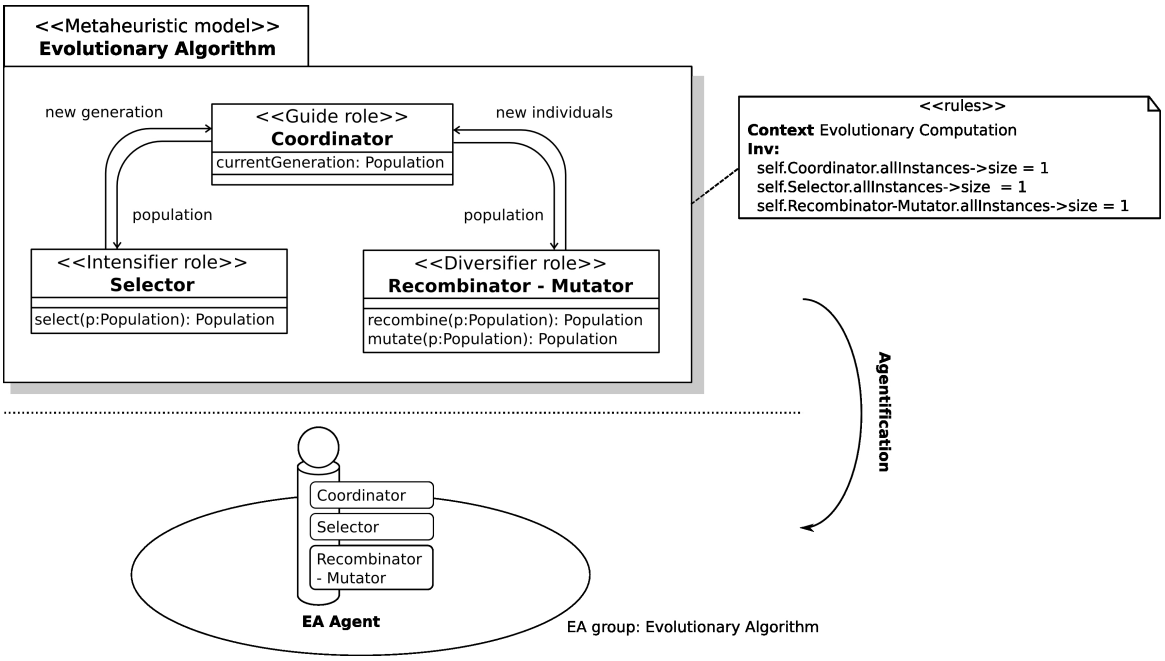

Figure 4.5: Un modèle organisationnel d'algorithme évolutionniste et son agentification

Les algorithmes évolutionnistes sont sûrement les méthodes où le plus grand nombre de variantes adaptatives ont été développées. Une revue des différents mécanismes d'adaptation est proposée dans [Hinterding et al., 1997] et [Eiben et al., 1999]. Outre la possibilité d'adapter la représentation des individus [Shaefer, 1987] ou encore le type d'opérateur, les principales variantes adaptatives des algorithmes évolutionnistes s'intéressent à modifier les paramètres stratégiques. Parmi ces paramètres, nous pouvons citer les taux de mutation et de croisement ainsi que la taille de la population. Ainsi, sans rentrer dans les détails des mécanismes d'adaptation, le rôle Stratège peut consister dans les algorithmes évolutionnistes à observer l'évolution des différentes générations de populations et à ajuster les paramètres stratégiques des rôles *Recombinator - Mutator*, *Selector* et *Coordinator*. Le modèle organisationnel de la figure 4.5 correspond à une version standard d'un algorithme évolutionniste, sans adaptation et donc sans Stratège.

Dans la partie inférieure de la figure 4.5 une agentification du modèle organisationnel est proposée. Le système obtenu est composé d'un seul agent jouant l'ensemble des rôles.

Il s'agit d'une version non distribuée d'un algorithme évolutionniste.

Il existe différentes approches de distribution des algorithmes évolutionnistes [Alba and Tomassini, 2002]. La figure 4.6 illustre l'approche des îles (*island EA*) [Tanese, 1989]. Cette approche de distribution, aussi qualifiée de "*coarse grain*" (à forte granularité), consiste en un ensemble de sous-populations séparées les unes des autres et ne communiquant que périodiquement au travers de migrations d'individus.

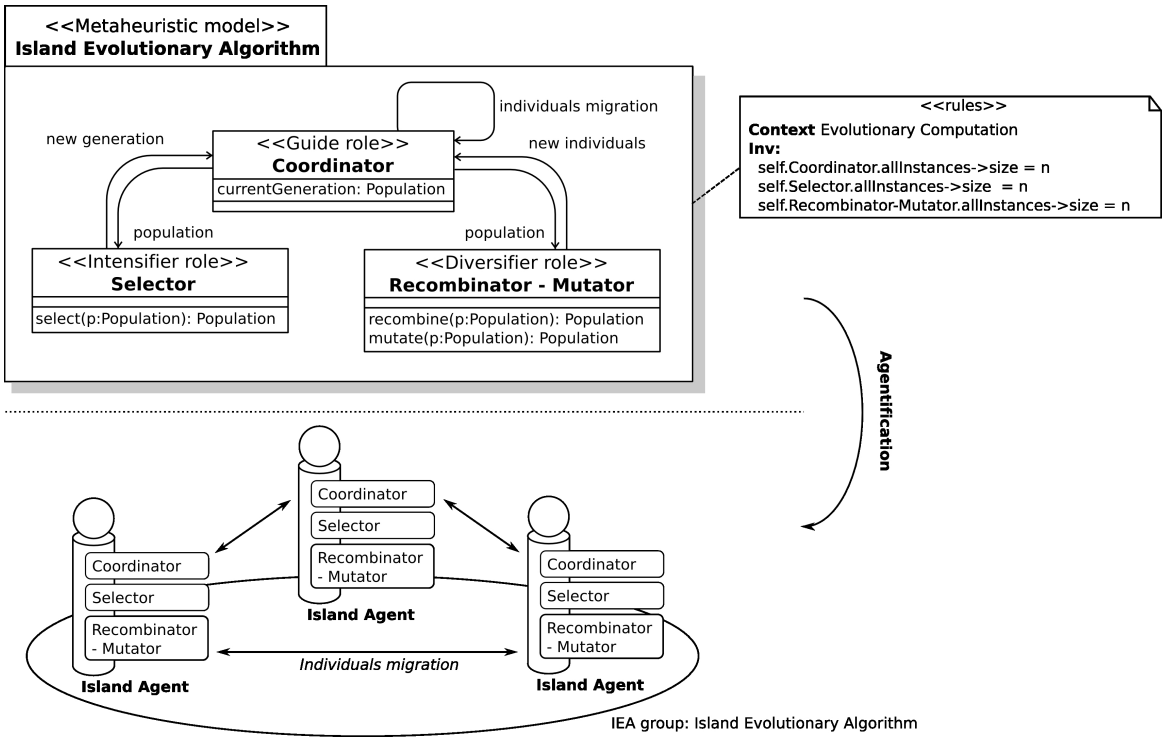

Figure 4.6: Un modèle organisationnel d'algorithme évolutionniste et son agentification suivant l'approche des îles

Pour modéliser cette variante des algorithmes évolutionnistes, le modèle organisationnel est légèrement adapté afin d'ajouter l'interaction correspondant à la migrations d'individus. Cette interaction est représentée dans le modèle organisationnel de la figure 4.6 au niveau du rôle *Coordinator*. De plus, les contraintes liées à l'instanciation des rôles ont été modifiées de manière à pouvoir instancier plusieurs fois les mêmes rôles. La partie inférieure de la figure 4.6, est une instance d'algorithme évolutionniste distribuée à la manière de l'approche des îles et composée de trois agents. Chaque agent joue l'ensemble des rôles et dispose donc d'une population d'individus. Les agents interagissent lors des migrations d'individus.

4.5.3 Optimisation par colonies de fourmis

L'optimisation par colonies de fourmis est inspirée du comportement collectif des fourmis. Ce comportement permet aux fourmis d'optimiser le chemin entre la nourriture et la

fourmilière. Chaque fourmi construit un chemin et dépose une trace de phéromone qui sera utilisée par les fourmis suivantes. Grâce aux dépôts successifs de phéromone et au phénomène d'évaporation, les fourmis empruntent progressivement un chemin plus court.

Dans l'optimisation par colonies de fourmis, la recherche de nouvelles solutions est effectuée par les fourmis. La mémoire qui guide cette recherche correspond à une matrice de phéromone. Celle-ci est utilisée par les fourmis pour construire de manière incrémentale les solutions. Chaque solution correspond à un ensemble de composantes qui ont été choisies par une fourmi. Lors du choix d'une composante de solution, la fourmi est soumise (i) à une tendance à exploiter les informations de la matrice de phéromone, mais aussi (ii) à une tendance à utiliser des informations heuristiques [Dorigo and Stützle, 2000]. Ces deux tendances sont combinées dans le processus de décision stochastique de la fourmi et correspondent respectivement à l'intensification et la diversification de la recherche.

Après avoir créé une solution, un dépôt de phéromone dont l'intensité dépend de la qualité de la solution est effectué. Les dépôts successifs de phéromone combinés à l'évaporation vont conduire les fourmis à trouver des solutions de meilleure qualité.

Pour obtenir un modèle organisationnel de cette métaheuristique, il est difficile de distinguer un comportement d'intensification d'un comportement de diversification, même si les deux tendances sont identifiables. Ainsi, dans le modèle organisationnel de l'optimisation par colonies de fourmis proposé, un unique rôle combine les rôles *Intensifier* et *Diversifier*.

Dans la partie supérieure de la figure 4.7 un modèle organisationnel de l'optimisation par colonies de fourmis est donné. La description des rôles composant l'organisation est la suivante :

Rôle *Ant* : L'intensification et la diversification sont combinées dans le processus de décision stochastique d'une fourmi. Ainsi, dans l'organisation *Ant Colony* le rôle *Ant* correspond à ces deux tendances. L'objectif de ce rôle est de construire de nouvelles solutions en utilisant la matrice de phéromone.

Rôle *Pheromone manager* : Ce rôle coordonne la recherche effectuée par les fourmis et gère une mémoire constituée d'une matrice de phéromone. Il peut être vu comme un raffinement du rôle Guide. Son comportement consiste à mettre à jour la matrice de phéromone lorsqu'une solution a été obtenue par une fourmi et à faire évaporer progressivement la phéromone.

Assez peu de versions adaptatives des algorithmes de colonies de fourmis ont été développées. Dans [Pilat and White, 2002], les auteurs proposent d'utiliser un algorithme génétique chargé de déterminer les meilleurs paramètres. Une autre approche nommée SAACO (*Self-Adaptive Ant Colony Optimization*) décrite dans [Randall, 2004] et [Förster et al., 2007] consiste à utiliser une matrice de phéromone relative aux paramètres stratégiques. Dans ces deux approches, l'objectif est d'ajuster les paramètres stratégiques tels que, le taux d'évaporation, la taille de la colonie de fourmis et les paramètres du modèle de décision des

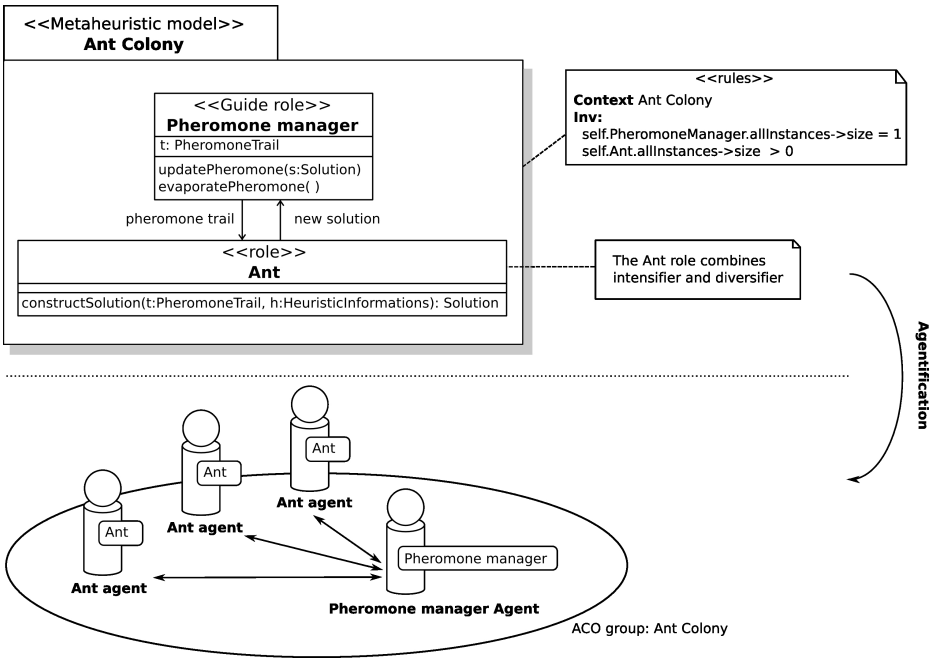

Figure 4.7: Modèle organisationnel de l'optimisation par colonies de fourmis

fourmis. Dans la première approche, fondée sur l'utilisation d'un algorithme génétique, le rôle Stratège consisterait à mettre en concurrence différentes colonies de fourmis ayant des paramètres propres. Dans la seconde approche, le rôle Stratège serait un processus combiné à l'algorithme de colonies de fourmis qui fait évoluer la matrice de phéromone relative aux paramètres. Cette seconde approche correspond à de l'auto-adaptation.

L'instanciation de l'organisation *Ant colony* présentée dans la partie inférieure de la figure 4.7 correspond à une distribution "naturelle" des colonies de fourmis. Dans cette distribution, un ensemble d'agents jouant le rôle *Ant* interagit avec un unique agent associé au rôle *Phéromone manager*.

4.5.4 Autres métaheuristiques

À travers ces trois exemples, nous avons montré comment utiliser le modèle organisationnel pour décrire différentes métaheuristiques. Le tableau 4.1 récapitule le raffinement des rôles du modèle organisationnel d'*AMF* pour plusieurs autres métaheuristiques. La première colonne du tableau indique le nom de la métaheuristique. Les trois colonnes suivantes correspondent aux raffinements des rôles Intensifieur, Diversifieur et Guide. La dernière colonne du tableau fait référence quelques possibilités de raffinement du rôle Stratège. Il s'agit d'une liste non exhaustive d'extensions adaptatives des métaheuristiques.

Il est possible d'identifier dans le tableau 4.1 plusieurs similitudes dans le raffinement des rôles Intensifieur, Diversifieur, Guide et Stratège. Tout d'abord, le rôle Intensifieur est principalement réalisé par la recherche locale (*ILS, SA, VNS, GLS, TS*) ou par la sélection (*EA, MA, SS, EDA*). Le rôle Diversifieur correspond, soit à une perturbation ou mouve-

ment aléatoire (*ILS, SA*), soit à une contrainte appliquée à la recherche locale (liste tabou *TS*, changement de voisinage *VNS* ou contrainte sur la fonction objectif *GLS*), ou encore la génération de nouvelles solutions (*AE, MA, SS, EDA*). Pour l'ensemble de ces métaheuristiques, le rôle guide consiste à coordonner les rôles Intensifieur et Diversifieur et à maintenir à jour la mémoire. Enfin, on distingue au niveau du rôle Stratège différentes manières d'évaluer les performances de la recherche, soit en mettant en compétition différentes stratégies de recherche (ex : approche *SAGA* [Hinterding et al., 1996]), soit en s'appuyant sur les expériences d'un unique processus de recherche (ex : approche *ASA* [Ingber, 1996]).

Table 4.1: Réalisation des rôles d'*AMF* pour différentes métaheuristiques

	Intensifieur	Diversifieur	Guide	Stratège
Recherche locale itérée (ILS)	Recherche locale (descente)	Perturbation	Mise à jour de la solution courante avec critère d'acception	Modification de l'intensité de perturbation ([Lourenço et al., 2003], [Battiti and Protasi, 1996])
Recuit simulé (SA)	Déplacement dans le voisinage de la solution		Mise à jour de la meilleure solution et de la température (refroidissement et recuit)	Modification de la loi de refroidissement et des paramètres du critère d'acceptation (approche ASA, Adaptive Simulated Annealing, [Ingber, 1996])
	Acceptation de meilleure solutions	Tendance à effectuer un mouvement non bénéfique		
Recherche à voisinage variable (VNS)	Recherche locale (descente)	Application de "secousse", changement de voisinage	Mise à jour de la meilleure solution	Modification de l'ordre des voisinage (SAVND, Self-Adaptive Variable Neighborhood Descent, [Hu and Raidl, 2006])
Recherche locale guidée (GLS)	Recherche locale (descente)	Modification des pénalités	Mise à jour de la meilleure solution et des pénalités	
Recherche tabou (TS)	Recherche locale contrainte		Mise à jour de la solution courante et de la liste tabou	Modification de la taille de la liste tabou (approche RTS, Reative Tabu Search, [Battiti and Tecchiolli, 1994])
	Tendance au choix de la meilleure solution dans le voisinage	Contraintes de la liste tabou		
Algorithme évolutionniste (EA)	Sélection	Mutation et croisement	Mise à jour de la population	Modification de la taille de population et des taux de mutation et croisement ([Hinterding et al., 1997], [Eiben et al., 1999])

Table 4.2: Réalisation des rôles d'*AMF* pour différentes métaheuristiques (Suite)

	Intensifieur	Diversifieur	Guide	Stratège
Algorithme mémétique (MA)	Sélection et recherche locale	Mutation et croisement	Mise à jour de la population	
Recherche dispersée (SS)	Sélection des meilleures solutions et recherche locale	Combinaison	Mise à jour de l'ensemble de référence	
Algorithme à estimation de distribution (EDA)	Échantillonnage		Mise à jour de l'estimation de distribution	
	Tendance à échantillonner dans les zones à meilleure estimation et sélection des solutions	Tendance aléatoire de l'échantillonnage		
Optimisation par colonies de fourmis (ACO)	Création de solutions par les fourmis		Mise à jour de la matrice de phéromone (dépôt de phéromone et évaporation)	Modification du taux d'évaporation, de la taille de la colonie et des paramètres de décision des fourmis ([Pilat and White, 2002], [Randall, 2004], [Förster et al., 2007])
	Tendance à suivre les traces de phéromone	Tendance aléatoire et à suivre les informations heuristiques		
Optimisation par essaim particulaire (PSO)	Déplacement des particules		Information mutuelle des particules	Modification de la taille de l'essaim et des paramètres de déplacement (approche TRIBES, [Clerc, 2005])
	Attraction des particules vers la meilleure solution trouvée et la meilleure connue	Inertie		

4.6 Conclusion

Dans ce chapitre, nous avons introduit le framework *AMF* destiné à l'analyse et à la conception de métaheuristiques. Le modèle organisationnel d'*AMF*, qui est à la base de ce framework, adopte les concepts du méta-modèle RIO et présente ainsi une métaheuristique sous la forme d'une organisation composée de quatre rôles en interaction. Ce modèle peut être considéré comme une trame ou un schéma qui doit être raffiné en fonction des caractéristiques particulières de la métaheuristique que nous voulons retenir, et suivant le ou les problèmes à traiter.

Le premier objectif recherché avec *AMF* est de donner un cadre d'analyse et de comparaison des différentes métaheuristiques existantes. Pour cela, *AMF* introduit de nouveaux critères de comparaison et d'analyse des métaheuristiques suivant l'occurrence des différents rôles du modèle organisationnel dans ces méthodes. Afin d'illustrer ces points, nous avons présenté comment modéliser plusieurs métaheuristiques existantes en utilisant notre approche.

Le second objectif d'*AMF* est de faciliter la conception de nouveaux algorithmes en encourageant une approche multi-agent. Sur ce point, la décomposition d'une métaheuristique en différents rôles vise à faciliter l'hybridation de différents aspects provenant de différentes métaheuristiques. De même, cela permet de dissocier la démarche de spécification, des aspects liés à la distribution sur les agents. La mise en application de cette démarche est l'objet du chapitre suivant où une métaheuristique multi-agent est proposée. Sa modélisation est réalisée suivant *AMF* et le modèle d'agents retenu doit incorporer les différents avantages inhérents à ce type de systèmes, notamment la résolution distribuée du problème et l'application de techniques d'apprentissage.

Les principales différences du framework *AMF* avec les frameworks de métaheuristiques existants (présentés dans la section 2.5) sont, tout d'abord, de rompre avec la vision séquentielle d'une métaheuristique présente par exemple dans les frameworks *AMP* et *ALS*. De même, et c'est ce qui nous différencie d'une approche telle que *MAGMA* où chaque couche représente une fonction particulière associée à un ou plusieurs agents, la distribution des rôles aux agents dans notre framework est basée sur la possibilité d'une résolution collective, dans laquelle la coopération tient un rôle essentiel pour expliquer et amplifier la performance. Enfin, le framework *AMF* intègre les concepts liés aux mécanismes d'adaptation et d'auto-adaptation dans les métaheuristiques.

CHAPITRE 5

CONCEPTION D'UNE MÉTAHEURISTIQUE À BASE DE COALITION D'AGENTS

Sommaire

5.1 Introduction

Nous proposons dans ce chapitre une métaheuristique multi-agent nommée *CBM* (*Coalition-Based Metaheuristic*). Cette métaheuristique est fondée sur la métaphore de la coalition. Dans *CBM*, chaque agent est capable d'effectuer indépendamment une recherche et adapte son comportement par apprentissage. Les agents sont organisés en coalition et coopèrent afin d'améliorer la recherche de solution. L'objectif de cette métaheuristique est d'exploiter les différents aspects des systèmes multi-agents pour concevoir une méthode efficace, robuste et modulaire.

Le premier aspect des systèmes multi-agents que nous souhaitons exploiter dans cette approche est la distribution conceptuelle du contrôle inhérente à ce type d'approche. Elle permet d'envisager une exécution parallèle de la métaheuristique, mais aussi de mettre en place des mécanismes de coopération entre agents. Ensuite, nous souhaitons intégrer des mécanismes d'apprentissage qui adaptent dynamiquement la stratégie de recherche afin de rendre la méthode plus robuste face à différentes instances d'un problème.

Pour concevoir cette métaheuristique, nous utilisons le framework *AMF*. Ainsi, en partant du modèle organisationnel de métaheuristique d'*AMF*, les règles méthodologiques sont suivies afin de modéliser *CBM* sous la forme d'un système multi-agent.

Dans ce chapitre, nous présentons tout d'abord le principe général de la métaheuristique *CBM*, puis nous détaillons le modèle organisationnel associé. Étant donné l'importance du principe de sélection d'opérateurs et des mécanismes d'apprentissage, nous présentons en détail ces deux points dans la section 5.4. Enfin, la section 5.5 décrit l'agentification du modèle organisationnel de *CBM*.

5.2 Principe de *CBM*

Le principe opératoire de *CBM* au niveau d'un agent individuel s'inspire de l'approche hyper-heuristique [Meignan et al., 2008c] introduite dans la section 3.4. Cette approche consiste à distinguer une couche stratégique, d'une couche de mise en œuvre de la recherche. Cette séparation permet notamment de concevoir des métaheuristiques modulaires. Dans cette section, nous interprétons et donnons une extension de l'approche hyper-heuristique en nous basant sur le modèle organisationnel d'*AMF*. Ensuite, nous présentons comment nous répartissons le processus d'optimisation sur une coalition d'agents et l'intérêt d'utiliser une telle structure.

5.2.1 Transposition de l'approche hyper-heuristique

Les hyper-heuristiques, peuvent être définies comme étant des méthodes d'optimisation ayant la particularité d'utiliser un mécanisme de sélection dynamique des heuristiques qui

vont résoudre un problème d'optimisation [Burke et al., 2003]. Dans ce cadre, les heuristiques sont des boîtes noires permettant de mettre en œuvre une recherche. Elles constituent la couche basse du processus d'optimisation. L'hyper-heuristique est la couche supérieure qui sélectionne les heuristiques à appliquer en fonction d'informations non spécifiques au problème traité. Par exemple, le choix d'une heuristique peut être conditionné par le temps d'exécution moyen de cette dernière, ou encore, par la différence de fitness apportée en moyenne par l'heuristique lors de ses précédentes applications. Le mécanisme de sélection est indépendant du problème. Par conséquent, la couche hyper-heuristique ne nécessite pas de spécialisation. Il est possible de traiter d'autres problèmes en remplaçant uniquement l'ensemble des heuristiques de la couche basse. Ce principe de séparation entre un niveau hyper-heuristique indépendant du domaine, et un niveau dépendant du domaine, est illustré dans la partie gauche de la figure 5.1. Dans la partie droite, est précisée la structure organisationnelle de *CBM*.

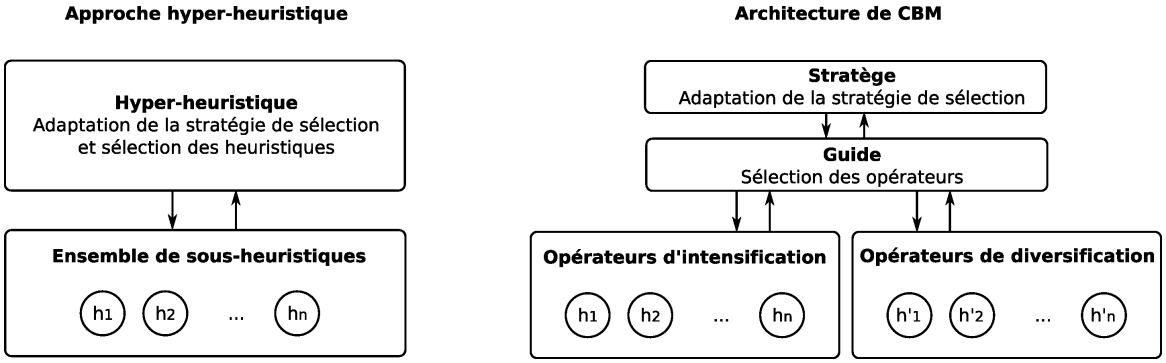

Figure 5.1: Schéma de l'architecture d'une hyper-heuristique (inspiré de [Burke et al., 2003]) et de l'architecture utilisée dans *CBM*

À partir de la définition que nous venons de donner des hyper-heuristiques, on remarque que les différents composants peuvent être décrits dans le modèle organisationnel d'*AMF*. En effet, la couche basse, constituée des heuristiques qui réalisent la recherche, correspond aux rôles Intensifieur et Diversifieur. Ensuite, le processus de sélection d'heuristique se rapporte au rôle Guide. Enfin, le mécanisme d'apprentissage qui évalue et ajuste la stratégie de sélection correspond au rôle Stratège. On remarque dans la structure en deux niveaux des hyper-heuristiques, que le mécanisme de contrôle des opérateurs (relatif au rôle Guide dans *AMF*) est confondu avec le mécanisme d'adaptation (relatif au rôle Stratège dans *AMF*). Dans notre cas, en rajoutant un troisième niveau, nous distinguons clairement le procédé d'adaptation, du mécanisme de sélection d'opérateurs. Nous faisons cette séparation dans notre modèle pour le rendre plus générique. Il englobe ainsi certaines versions rudimentaires des méthodes dites à voisinage variables *Adaptive Large Neighborhood Search* (ALNS) [Ropke and Pisinger, 2006] par un simple retrait du rôle Stratège.

La métaheuristique que nous présentons dans la suite de ce chapitre reste cependant assez proche des hyper-heuristiques. En effet, dans *CBM* plusieurs heuristiques sont coordon-

nées par un processus de décision non spécifique au problème traité. Ainsi, nous souhaitons exploiter les deux principaux avantages de l'approche hyper-heuristique correspondant à la combinaison d'heuristiques et la généricité du mécanisme de sélection. En plus d'étendre la structure des hyper-heuristiques en ajoutant un niveau supplémentaire, la principale originalité de *CBM* est sa potentialité à être mise en œuvre de manière distribuée. Elle permet, d'améliorer l'exploration de l'espace de recherche, et de mettre en concurrence plusieurs stratégies de recherche. Dans notre approche, l'optimisation est distribuée sur une coalition d'agents où chaque agent implante l'architecture que nous venons de présenter.

5.2.2 Structure générale de *CBM*

CBM est une métaheuristique fondée sur la métaphore de la coalition. Une coalition est une structure d'organisation où les acteurs sont réunis pour résoudre un même problème. Dans le domaine des systèmes multi-agents, cette structure présente des caractéristiques particulières. Dans Horling and Lesser [2005], les auteurs définissent une coalition comme une structure organisationnelle plate, c'est-à-dire sans hiérarchie explicite. Au sein d'une telle structure, les agents coordonnent leurs actions, et cette coopération participe à améliorer l'efficacité de leurs tâches individuelles ou de l'activité globale de la coalition [Horling and Lesser, 2005]. Dans [Parunak et al., 2003], les auteurs définissent une coalition comme un système utilisant conjointement la conversation (interaction directe de type pair-à-pair) et la coopération (intention commune de réaliser un objectif). Ces deux propriétés impliquent une collaboration entre agents. Il en résulte une coalition d'agents. En résumé, une coalition est une structure sans hiérarchie, où les agents coopèrent par le biais d'interactions directes.

L'intérêt d'une telle structure est en premier lieu la décentralisation. Cette caractéristique joue en faveur de la robustesse du système lors de l'ajout ou de la suppression d'agents, notamment par le fait qu'aucun agent en particulier ne joue un rôle tel que sa suppression conduirait à une panne irrémédiable du système. De plus, l'utilisation d'interactions directes facilite la mise en œuvre parallèle. *CBM* exploite ces deux aspects de la structure de coalition.

Dans *CBM*, l'optimisation est réalisée par un ensemble d'agents ayant les mêmes fonctions et traitant la même instance de problème. Chaque agent est capable d'effectuer indépendamment des autres une recherche de solution. Cependant, la coopération entre les agents permet d'améliorer l'efficacité de l'optimisation. La coopération est réalisée grâce à l'échange d'informations sur les zones prometteuses de l'espace de recherche, et l'échange d'informations relatives aux stratégies de recherche performantes.

Un agent au sein de *CBM* dispose d'une solution courante qu'il fait évoluer à l'aide d'un ensemble d'opérateurs. Ces opérateurs sont des procédures d'intensification ou de diversification. Le choix d'un opérateur est réalisé par un processus de décision qui définit la stratégie de recherche de l'agent. Il permet d'effectuer le choix de l'opérateur à appliquer

en fonction du contexte d'optimisation. Le terme "contexte" fait référence à l'ensemble des données à disposition de l'agent, lui permettant de prendre une décision. Il s'agit par exemple, des caractéristiques de la solution courante ou de la connaissance des opérateurs ayant été appliqués récemment.

La stratégie définie dans le cadre du système de décision est adaptée dynamiquement par des mécanismes d'apprentissage individuels et collectifs. Au niveau de chaque agent, un processus d'apprentissage par renforcement permet d'ajuster la stratégie de recherche en fonction des expériences propres de l'agent. Ce premier mécanisme d'apprentissage est combiné avec un mécanisme d'apprentissage par mimétisme. Ce type d'apprentissage permet à un agent d'imiter les stratégies efficaces de recherche des autres agents.

Ainsi les principales caractéristiques de l'approche *CBM* sont, (i) la distribution de la recherche sur une coalition d'agents, (ii) l'utilisation au niveau de chaque agent d'un mécanisme de décision permettant de conditionner le choix d'un opérateur au contexte d'optimisation, et (iii) l'utilisation combinée d'un mécanisme d'apprentissage par renforcement avec un mécanisme par mimétisme pour améliorer la stratégie de recherche.

5.3 Modèle organisationnel de *CBM*

La description et la modélisation de *CBM* sont basées sur le framework *AMF*. Dans cette section le modèle organisationnel d'*AMF* est raffiné afin d'obtenir un modèle de l'organisation de *CBM*. Ce raffinement correspond à la première phase décrite par le guide méthodologique présenté dans la section 4.4 et permet de définir le comportement ainsi que les interactions associés aux rôles Intensifieur, Diversifieur, Guide et Stratège. Pour cette phase de raffinement, la structure du système multi-agent n'est pas encore définie. L'objectif est d'obtenir un modèle organisationnel qui en principe peut être agentifié de différentes manières.

5.3.1 L'organisation *CBM*

Dans l'organisation relative à *CBM*, les quatre rôles du modèle organisationnel d'*AMF* sont présents. Ces rôles nommés *Improver*, *Explorer*, *Manager* et *Learner* sont les raffinements respectifs des rôles *Intensifier*, *Diversifier*, *Guide* et *Strategist*. La figure 5.2 présente le modèle organisationnel de *CBM* constitué de ces quatre rôles. Pour chacun des rôles, les principaux attributs et méthodes sont décrits.

La recherche est effectuée par un ensemble d'opérateurs de diversification et d'intensification. Ici, un opérateur est une procédure (génération, recherche locale, mutation, etc.) permettant d'obtenir une solution à partir d'un ensemble éventuellement vide de solutions. Ces opérateurs sont associés respectivement aux rôles *Improver*, lorsqu'ils consistent à améliorer les solutions, et *Explorer*, lorsqu'ils permettent d'explorer l'espace de recherche.

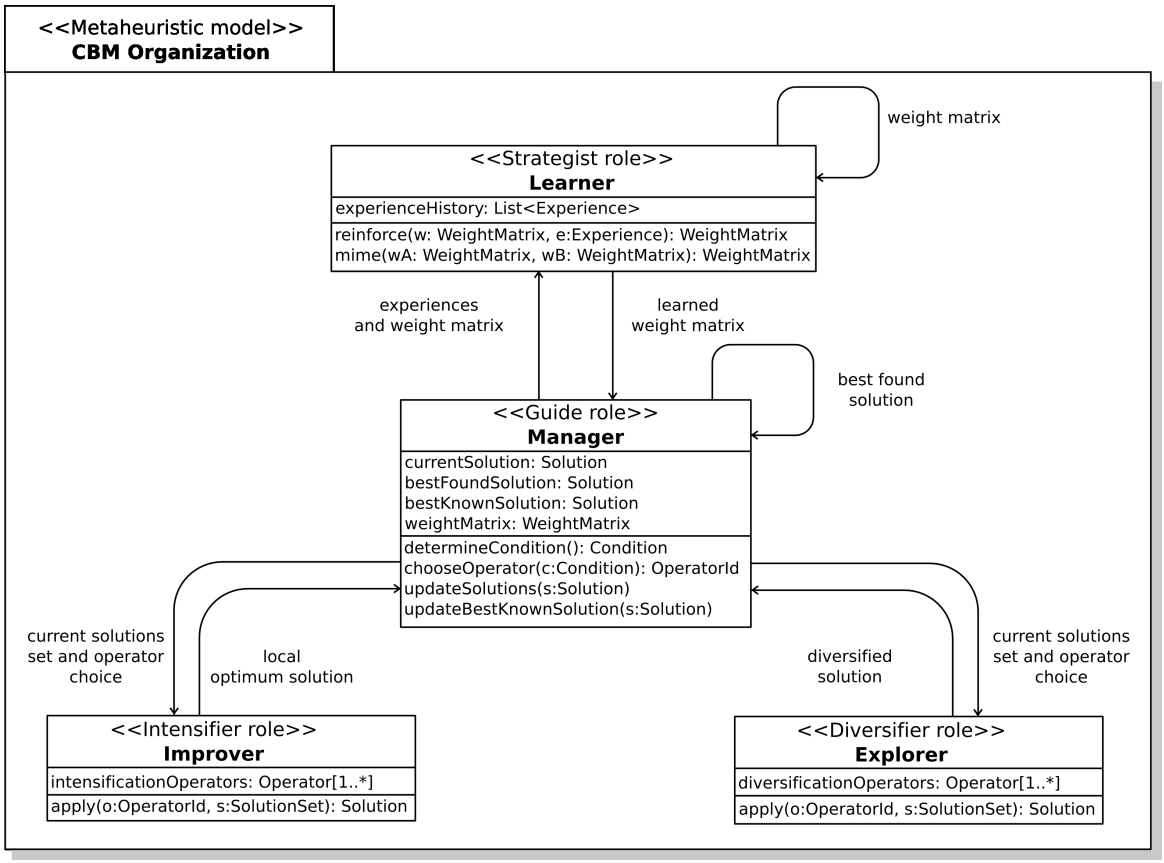

Figure 5.2: Modèle organisationnel de *CBM*

L'objectif est de regrouper le plus grand nombre d'opérateurs simples pour qu'ils soient automatiquement combinés.

L'exécution des différents opérateurs est coordonnée par un mécanisme de décision permettant de choisir l'opérateur à appliquer en fonction du contexte d'optimisation. Ce mécanisme de décision est associé au rôle *Manager*, raffinement du rôle *Guide*. Le rôle *Manager* est chargé d'appliquer une stratégie de recherche et de maintenir à jour une mémoire constituée des meilleures solutions. La stratégie de recherche suivie par le rôle *Manager* est adaptée dynamiquement par le rôle *Learner*, raffinement du rôle *Strategist*. L'adaptation est réalisée suivant un apprentissage par renforcement et un apprentissage par mimétisme. Chaque rôle est détaillé dans les sections suivantes.

5.3.2 Rôle *Improver*

Le raffinement du rôle Intensifieur dans l'organisation *CBM* est nommé *Improver*. Ce rôle utilise un ensemble d'opérateurs de recherche locale permettant d'obtenir un minimum local à partir d'une solution. Cette liste d'opérateurs est représentée par l'attribut *intensificationOperators* dans la figure 5.2. Le comportement du rôle Intensifieur consiste à appliquer une recherche locale donnée, à une solution qu'on lui a soumise.

5.3.3 Rôle *Explorer*

Le rôle Diversifieur est nommé *Explorer* dans l'organisation *CBM*. Ce rôle dispose d'un ensemble d'opérateurs de génération, mutation ou croisement de solutions représenté par l'attribut *diversificationOperators* dans la figure 5.2. Un opérateur de génération est une procédure qui permet de construire une nouvelle solution. Les opérateurs de mutation appliquent une perturbation à une solution et les opérateurs de croisement combinent les composantes de deux solutions. Dans le cas de *CBM*, les opérateurs de croisement sont considérés comme des opérateurs de diversification étant donné que la combinaison des composantes des deux solutions est réalisée aléatoirement et peut entraîner une redirection vers une nouvelle zone de l'espace de recherche. Le comportement du rôle Diversifieur consiste à appliquer un opérateur de diversification donné sur un ensemble de solutions afin de produire une nouvelle solution. L'ensemble des solutions fournies en entrée au rôle est vide lorsque l'opérateur spécifié est un opérateur de génération, il contient une unique solution lorsqu'il s'agit d'un opérateur de mutation, et il contient deux solutions s'il s'agit d'un opérateur de croisement.

5.3.4 Rôle *Manager*

Dans l'organisation *CBM*, le rôle *Guide* est raffiné en un rôle nommé *Manager*. Ce rôle gère une mémoire constituée de trois solutions, similaire à la mémoire des particules dans l'optimisation par essaims particulaires [Clerc, 2005]. La première solution est la solution courante qui évolue via l'application des opérateurs d'intensification et de diversification gérés par les rôles *Improver* et *Explorer*. La seconde solution est la meilleure solution trouvée lors de l'évolution de la solution courante. La dernière solution correspond à la meilleure solution connue, obtenue soit lors de la recherche, soit par interaction avec les autres instances du rôle Guide.

Le comportement du rôle guide consiste, d'une part, à faire évoluer la solution courante, et d'autre part, à mettre à jour sa mémoire.

Pour faire évoluer la solution courante, un opérateur de diversification ou d'intensification est choisi en fonction du contexte d'optimisation. Cet opérateur est ensuite appliqué à la solution courante via le rôle *Improver* ou le rôle *Explorer*. Ainsi, la solution courante est continuellement soumise aux opérateurs, tant que le critère d'arrêt de l'optimisation n'est pas atteint. Si l'opérateur choisi est un opérateur de génération, alors aucune solution n'est envoyée à l'opérateur et la solution résultante remplace la solution courante. Dans le cas d'un opérateur de croisement, la solution courante et la meilleure solution connue sont soumises au rôle *Explorer* pour obtenir une nouvelle solution courante. Dans les autres cas, c'est la solution courante qui est soumise à l'opérateur.

Le choix d'un opérateur repose sur processus de décision. Ce processus de décision permet à partir d'une condition qui caractérise le contexte d'optimisation de choisir l'opé-

rateur à appliquer. Le contexte d'optimisation fait principalement référence à la mémoire sur laquelle peut être fondée une décision. Ce système de décision est fondé sur une matrice de poids représentée par l'attribut *weightMatrix* dans la figure 5.2. Le processus de décision est détaillé dans la suite du chapitre (section 5.4.1).

Après chaque application d'un opérateur, la mémoire est mise à jour. Si la nouvelle solution courante améliore la meilleure solution trouvée ou la meilleure solution connue alors celles-ci sont mises à jour. De plus, lors de l'obtention d'une nouvelle meilleure solution connue par une instance du rôle *Manager*, cette solution est transmise aux autres instances du rôle *Manager*.

5.3.5 Rôle *Learner*

La stratégie de recherche mise en oeuvre par le rôle *Manager* est dictée par un mécanisme de décision. Dans *CBM*, cette stratégie est adaptée dynamiquement par le rôle *Learner*. Pour adapter la stratégie de recherche, deux mécanismes d'apprentissage sont combinés, un apprentissage par renforcement et un apprentissage par mimétisme. L'apprentissage par renforcement consiste à favoriser les actions ayant permis d'obtenir de bonnes solutions. L'apprentissage par mimétisme consiste à observer les autres rôles stratèges et à tirer parti des stratégies efficaces pour modifier les règles de décision. Ces deux types d'apprentissage sont détaillés dans la section 5.4.2.

5.4 Processus de décision et apprentissage dans *CBM*

5.4.1 Processus de décision

L'objectif du système de décision dans *CBM* est de sélectionner l'opérateur à appliquer en fonction de ce que nous avons appelé le contexte d'optimisation. Ce processus utilisé par le rôle *Manager* permet de coordonner l'intensification et la diversification. Il permet des recherches à voisinage variable à la manière des approches de type *Adaptive Large Neighborhood Search* (ALNS) [Ropke and Pisinger, 2006] et peut présenter des similarités avec les systèmes de classifieurs de Holland [Holland et al., 2000].

Dans l'approche *ALNS* (*Adaptive Large Neighborhood Search*), les auteurs proposent une méthode de résolution fondée sur la combinaison de plusieurs sous-heuristiques. Un poids est associé à chacune des heuristiques et un mécanisme de sélection par roulette biaisée (*roulette wheel selection principle*) permet de choisir l'heuristique à appliquer en fonction de ce poids. Le vecteur de poids est ajusté automatiquement au cours de l'optimisation afin de favoriser les heuristiques les plus performantes. Dans [Ropke and Pisinger, 2005], les auteurs comparent les résultats de l'approche *ALNS* avec et sans le mécanisme d'adaptation dynamique de la sélection d'heuristiques. Cette étude expérimentale indique

que le mécanisme d'adaptation utilisé permet d'améliorer significativement les résultats de la méthode.

Dans *CBM*, le processus de décision permet de différentier les opérateurs efficaces des opérateurs secondaires à la manière de l'approche *ALNS*, mais aussi, de conditionner le choix des opérateurs au contexte d'optimisation. Le mécanisme de sélection d'action mis en œuvre ici est fondé sur une matrice de poids condition/action à partir de laquelle une sélection de l'opérateur est réalisée par un tirage aléatoire de type roulette biaisée.

Le mécanisme de sélection peut être précisé de la manière suivante. Soit C l'ensemble fini des conditions d'entrée du système de décision et O l'ensemble fini des opérateurs pouvant être sélectionnés. Pour chaque couple c_i, o_j un poids $w_{i,j}$ est défini. Ce poids permet de définir la probabilité $P(o_j|c_i)$ de sélectionner l'opérateur o_j sous la condition c_i suivant la formule ci-après.

$$P(o_j|c_i) = \frac{w_{i,j}}{\sum_{k=1}^{m} w_{i,k}} \tag{5.1}$$

Avec :

$C : (c_i)_{i=1,\ldots,n}$	Ensemble des conditions
$O : (o_j)_{j=1,\ldots,m}$	Ensemble des opérateurs
$W : (w_{i,j})_{i=1,\ldots,n;j=1,\ldots,m}$	Matrice de poids

Pour sélectionner un opérateur, le contexte d'optimisation est analysé afin de produire une condition d'entrée du système de décision. Un tirage de type roulette biaisée est ensuite effectué afin de sélectionner l'opérateur. La figure 5.3 représente un exemple de système de décision, permettant d'illustrer la fonction de sélection d'un opérateur. À gauche de la figure, les poids des couples condition/opérateur sont représentés sous forme d'une matrice, et à droite sous la forme d'un graphe pondéré. Les arcs de poids nul ne sont pas représentés étant donné qu'ils correspondent à des probabilités nulles de sélection d'opérateurs. Le système de décision représenté considère 4 conditions et 4 opérateurs et détermine les probabilités de sélection de chaque opérateur lorsqu'une condition est activée. Par exemple, si la condition c_1 est activée, l'opérateur o_1 a une probabilité de sélection de 75% et l'opérateur o_2 a une probabilité de sélection de 25%.

L'intérêt du processus de décision décrit ci-dessus réside dans sa simplicité de mise en œuvre et de représentation. À l'aide de ce système, il est possible de restreindre le choix des opérateurs en fonction du contexte d'optimisation en affectant des poids nuls, ou de favoriser un opérateur dans un contexte donné en augmentant le poids correspondant. Les mécanismes d'apprentissage ont pour tâche de modifier la matrice des poids au cours de la recherche et suivant l'expérience acquise.

L'ensemble des conditions a été choisi de manière à pouvoir réaliser une alternance entre la sélection d'opérateurs de diversification et d'opérateurs d'intensification. L'objectif est d'obtenir un cycle diversification/intensification. La première phase consiste en l'application d'un opérateur de diversification. La seconde phase consiste à appliquer les opérateurs

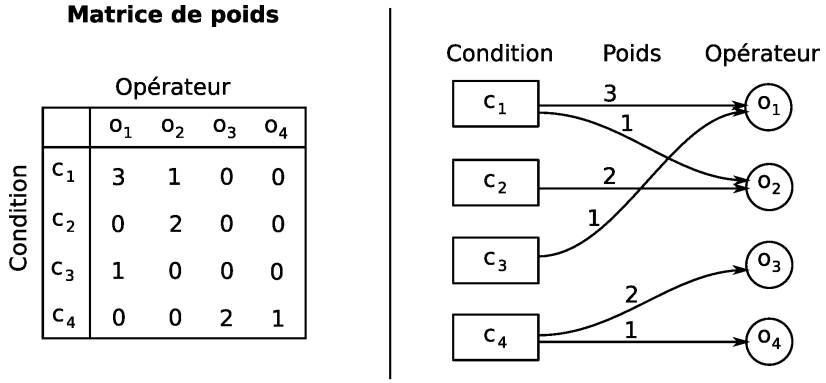

Figure 5.3: Exemple de matrice de poids pour un système de décision à 4 conditions et 4 opérateurs

d'intensification jusqu'à l'obtention d'un optimum local sur l'ensemble des structures de voisinages considérées. La définition de l'ensemble des conditions est un élément essentiel pour le bon fonctionnement du système de décision. Nous avons choisi ici de privilégier la prise en compte des ordonnancements possibles d'opérateurs, suivant qu'un opérateur donné est appliqué après l'autre. Pour ce faire, chaque condition est identifiée à l'application précédente d'un opérateur donné. Ce qui signifie notamment qu'un minimum local relativement à cet opérateur a été atteint.

Plus précisément, soit $I : (i_i)_{i=1,...,p}$ l'ensemble des opérateurs d'intensification et $D : (d_i)_{i=1,...,q}$ l'ensemble des opérateurs de diversification. Nous définissons trois types de conditions. La première c_0 est activée après l'application d'un opérateur de diversification. Les conditions c_1 à c_{n-2} sont respectivement obtenues suite à l'application des opérateurs d'intensification i_1 à i_p. La dernière condition c_{n-1} est activée une fois que la solution courante est un optimum local pour l'ensemble des opérateurs i_1 à i_p. Il faut noter, que la vérification de cette dernière condition doit tenir compte des précédentes applications d'opérateurs lorsque ceux-ci n'ont pas réussi à modifier la solution courante.

Prenons un exemple en considérant un système de décision avec deux opérateurs d'intensification (i_1 et i_2) et deux opérateurs de diversification (d_1 et d_2). Dans ce cas précis, quatres conditions serons considérées :

- $\mathbf{c}_0$ **:** Un opérateur de diversification vient d'être appliqué. La solution courante n'est potentiellement pas un optimum local.
- $\mathbf{c}_1$ **:** L'opérateur d'intensification i_1 vient d'être appliqué. La solution courante est un optimum local pour cet opérateur.
- $\mathbf{c}_2$ **:** L'opérateur d'intensification i_2 vient d'être appliqué. La solution courante est un optimum local pour cet opérateur.
- $\mathbf{c}_3$ **:** Les deux opérateurs d'intensification ont été appliqués à la suite l'un de l'autre sans modifier la solution courante. La solution courante est un optimum local pour l'ensemble des opérateurs d'intensification.

Au début de la recherche, les poids sont initialisés avec un paramètre α. Pour obtenir un

cycle diversification/intensification, plusieurs couples condition/opérateur sont initialisés à la valeur 0. Les deux règles régissant cette initialisation et permettant d'obtenir ce cycle, sont les suivantes : (i) Un opérateur de diversification ne peut être choisi que si la solution courante est un optimum local sur l'ensemble des structures de voisinage considérées. (ii) Un opérateur de recherche locale utilisant une structure de voisinage $\mathcal{N}_i$ ne peut être appliqué si la solution courante est un optimum local sur $\mathcal{N}_i$.

Pour illustrer cette initialisation, la figure 5.4 représente la matrice des poids pour un ensemble I d'opérateurs d'intensification et un ensemble D d'opérateurs de diversification. Une description de chaque condition est donnée dans la partie gauche de la figure. Les valeurs 0 et α placées dans la matrice correspondent à l'initialisation. Pour améliorer la lisibilité de la figure, toutes les valeurs de poids nulles ne sont pas indiquées. Les deux blocs qui encadrent des valeurs correspondent aux phases d'intensification et de diversification. La diversification ne concerne qu'une seule ligne étant donné que cette phase correspond à l'application d'un seul opérateur de diversification.

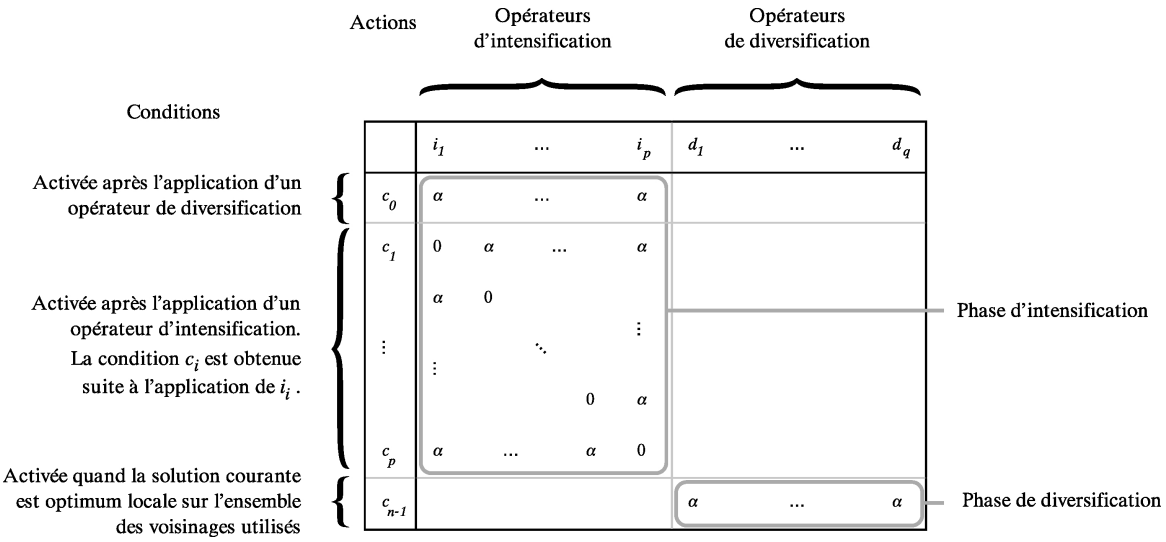

Figure 5.4: Cas général de l'initialisation de la matrice de décision

Même si le choix des opérateurs est limité par l'initialisation, en modifiant les poids non nuls de la matrice, il est possible de modifier la fréquence ainsi que l'ordre d'application des opérateurs. Cette modification des poids de la matrice est l'objet des mécanismes d'apprentissage présentés dans la section suivante.

5.4.2 Apprentissage

Afin d'adapter le comportement fondé sur le processus de décision précédemment formulé, deux mécanismes d'apprentissage sont utilisés conjointement : par renforcement et par mimétisme. L'objectif de l'apprentissage est d'ajuster la matrice de poids du système de décision afin d'améliorer le choix des opérateurs. Ces mécanismes d'apprentissage sont

à la base du rôle *Learner* de *CBM*.

Apprentissage par renforcement

Le problème de l'apprentissage par renforcement [Sutton and Barto, 1998] est un problème de sélection d'action dont l'objectif est de maximiser les récompenses engendrées par les décisions. Ce problème intervient lorsqu'un agent doit apprendre à choisir les actions à réaliser en observant les signaux ou les récompenses qu'il obtient en retour de ses décisions passées. Les points clés de ce problème sont la nécessité de chercher par essais/erreurs et le fait que les décisions peuvent avoir un effet retardé sur les récompenses. On appelle méthode ou mécanisme d'apprentissage par renforcement toute méthode adaptée à la résolution de ce problème.

Nous pouvons considérer la sélection d'opérateurs comme un problème d'apprentissage par renforcement. Ainsi, nous souhaitons mettre en place un mécanisme qui, en fonction des expériences de sélections d'opérateurs, modifie la matrice de poids pour la rendre plus efficace. Nous définissons une expérience comme étant un triplet ⟨condition, opérateur, gain⟩, où le gain est la différence de coût obtenue par l'application de l'opérateur. La méthode d'apprentissage que nous définissons est basée sur l'observation des expériences, et plus précisément sur l'observation des expériences relatives à un cycle diversification/intensification. Le renforcement est appliqué à l'issue de ces cycles.

Les règles d'apprentissage que nous avons définies sont les suivantes. Lorsqu'un cycle diversification/intensification a permis d'améliorer le coût de la meilleure solution trouvée, alors les poids correspondant aux expériences réalisées dans ce cycle sont augmentés. Deux valeurs de renforcement de poids sont définies σ_1 et σ_2. La valeur σ_1 est appliquée lorsque, dans le cycle diversification/intensification, seule la valeur de la meilleure solution trouvée est améliorée. La valeur σ_2 est appliquée si meilleure solution connue est améliorée. Le renforcement d'une expérience est donné par la formule 5.2.

$$w_{i,j} = w_{i,j} + \sigma \tag{5.2}$$

Avec :

$(c_i\,;\,o_j)$	Expérience à renforcer
$w_{i,j}$	Valeur de poids associée à l'expérience
$\sigma : \{\sigma_1; \sigma_2\}$	Valeur de la sanction

Pour illustrer ce mécanisme d'apprentissage, considérons un système de décision permettant de choisir un opérateur donné parmi deux opérateurs d'intensification o_1, o_2 et deux opérateurs de diversification o_3, o_4. En considérant que les opérateurs d'intensification sont des procédures de recherche locale utilisant des structures de voisinages distinctes, quatre conditions sont utilisées par le système de décision. La figure 5.5 présente un cas donnant lieu à renforcement. Le graphique dans la partie supérieure de la figure représente l'évolu-

tion de la fitness de la solution courante et celle de la meilleure solution trouvée. La fitness de la solution courante évolue suite à l'application de quatre opérateurs. Le premier, o_3, est un opérateur de diversification. La séquence d'opérateurs $[o_1, o_2, o_1, o_2]$ correspond à la phase d'intensification. La sélection de ces opérateurs est réalisée à l'aide de la matrice de poids initiale représentée en bas à gauche de la figure. Durant la phase d'intensification, la fitness de la meilleure solution trouvée est améliorée. Ainsi, à la fin du cycle diversification/intensification, les expériences dont le gain est non nul sont renforcées à l'aide de la valeur $\sigma = 1$ dans cet exemple. La matrice de poids après renforcement est représentée en bas à droite de la figure. Dans cette matrice, les poids associés aux couples condition/opérateur $\langle c_4, o_3 \rangle$, $\langle c_1, o_1 \rangle$ et $\langle c_2, o_2 \rangle$ ont été augmentés. Ce renforcement va influencer la sélection des prochains opérateurs. Par exemple, si la condition c_1 est activée, alors l'opérateur o_1 a plus de probabilité d'être choisi que l'opérateur o_2 avec respectivement 66% et 33% de probabilité de sélection.

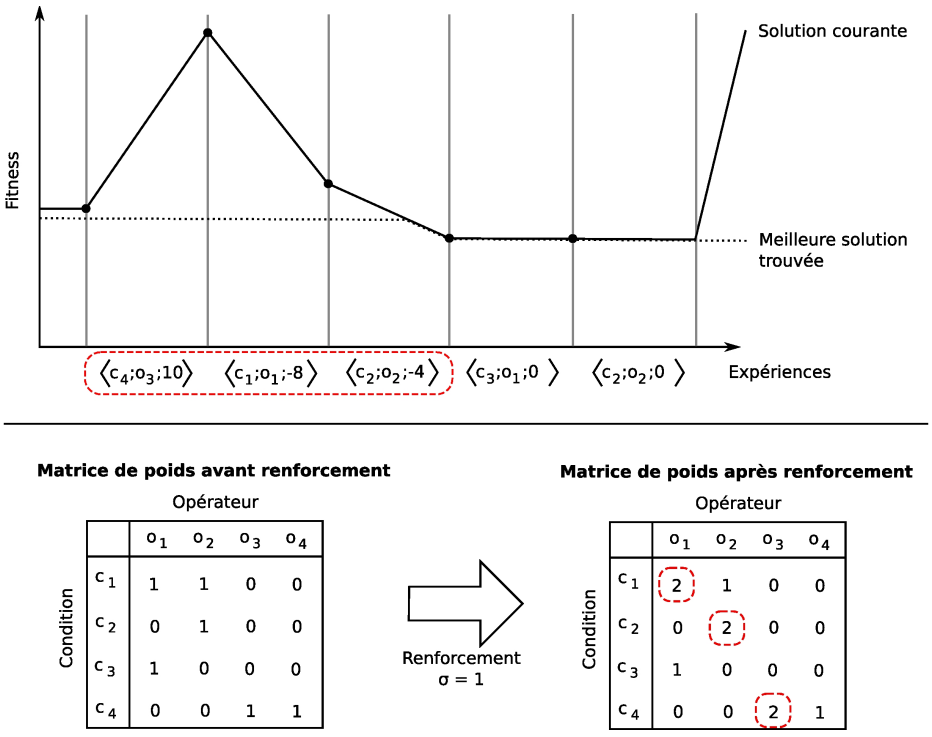

Figure 5.5: Exemple d'apprentissage par renforcement

Apprentissage par mimétisme

L'apprentissage par renforcement est réalisé individuellement dans chaque instance du rôle stratège. Le mimétisme permet de partager les connaissances individuelles acquises par renforcement. Le principe de l'apprentissage par mimétisme introduit par [Yamaguchi et al., 1997] consiste à imiter les comportements efficaces. Dans notre cas, un comportement est identifié comme efficace pour le mimétisme lorsqu'un cycle diversification/intensification a permis d'améliorer la meilleure solution connue.

Lorsque le rôle *Learner* observe une amélioration de la meilleure solution connue, il effectue un apprentissage par renforcement, puis soumet la matrice de poids aux autres rôles stratèges afin qu'ils en exploitent le contenu. Notons W_A la matrice de poids correspondant à la stratégie à imiter, et W_B la matrice de poids de l'imitateur. Nous définissons la matrice de poids résultant de l'imitation par une moyenne pondérée de W_A et W_B selon la formule 5.3.

$$W_B = (1 - \rho).W_B + \rho.W_A \tag{5.3}$$

Avec :

W_A	Matrice de poids à imiter
W_B	Matrice de poids de l'imitateur
ρ	Coefficient d'imitation

L'apprentissage par mimétisme est un mécanisme complémentaire à l'apprentissage par renforcement. En considérant que les matrices de poids sont initialisées à l'identique, l'apprentissage par mimétisme n'apporte aucune information supplémentaire s'il n'y a pas eu un apprentissage par renforcement préalablement. Mais du fait de la dynamique collective qu'il introduit, il peut notamment permettre de contrebalancer certains phénomènes de convergence prématurée de l'apprentissage par renforcement.

5.5 Agentification du modèle organisationnel de *CBM*

Nous avons défini dans la section précédente un modèle organisationnel de l'approche *CBM*. À partir de ce modèle, plusieurs structures de systèmes multi-agents peuvent être envisagées. Cependant, nous souhaitons obtenir un système décentralisé à la manière d'une coalition. Pour obtenir cette coalition, les agents jouent l'ensemble des rôles de l'organisation.

Chaque agent instancie l'ensemble des quatre rôles de l'organisation *CBM*. Ils héritent chacun d'une instance propre des propriétés relatives aux rôles. Un agent dispose donc d'une liste d'opérateurs d'intensification issue du rôle *Improver*, d'une liste d'opérateurs de diversification relative au rôle *Explorer*. De même, il dispose d'un ensemble de trois solutions (solution courante, meilleure solution trouvée et meilleure solution connue) et d'une matrice de poids issue du rôle *Manager* et d'un historique des expériences de recherche relatif au rôle *Learner*.

Le comportement d'un agent correspond à la combinaison des comportements des quatre rôles qui lui sont associés. Bien que ce comportement puisse être décrit au travers de différents processus qu'il faut ordonnancer, nous donnons ici une version séquentielle du comportement d'un agent. L'algorithme 4 présente le comportement d'un agent dans *CBM*. Ce comportement correspond à une version séquentielle d'un ordonnancement particulier des quatre rôles affectés à l'agent.

L'algorithme débute par l'initialisation des structures de données de l'agent. Ensuite, la boucle principale de l'algorithme s'exécute tant que le critère d'arrêt n'est pas atteint. Les critères d'arrêt que nous utilisons sont, soit un nombre limite d'application d'opérateurs, soit une durée maximale d'exécution. Lors d'une itération de la boucle principale, le rôle *Manager* consiste à sélectionner un opérateur (ligne 4), l'appliquer (ligne 5) et mettre à jour les solutions (ligne 6). Les deux blocs conditionnels suivants (ligne 7 à 12) consistent à envoyer et recevoir les nouvelles meilleures solutions connues. L'envoi d'une solution se fait vers l'ensemble des autres agents de la coalition. La fin de la boucle principale (ligne 13 à 21) correspond aux tâches attribuées au rôle *Learner*. L'apprentissage par renforcement et le partage de la matrice de poids pour le mimétisme ne s'effectuent qu'à la fin d'un cycle diversification/intensification. Si ce cycle a permis d'améliorer la meilleure solution trouvée, alors, les couples condition/opérateur relatifs au cycle diversification/intensification sont renforcés (ligne 14). Si le cycle a permis, en plus, d'améliorer la meilleure solution connue, alors la matrice de poids de l'agent est envoyée aux autres agents (ligne 16). Enfin, l'agent vérifie s'il a reçu une matrice de poids pour réaliser un apprentissage par mimétisme (ligne 19 à 21).

Algorithme 4 : Version séquentielle du comportement d'un agent dans *CBM*

```
1  initialisation des solutions
2  initialisation de la matrice de poids
3  tant que critère d'arrêt non atteint faire
4      choix de l'opérateur
5      application de l'opérateur
6      mise à jour des solutions
7      si nouvelle meilleure solution connue alors
8          envoi de la meilleure solution connue
9      fin
10     si réception d'une nouvelle meilleure solution connue alors
11         mise à jour de la meilleure solution connue
12     fin
13     si condition de renforcement alors
14         apprentissage par renforcement
15         si condition de mimétisme alors
16             Partage de la matrice de poids
17         fin
18     fin
19     si réception d'une matrice de poids alors
20         apprentissage par mimétisme
21     fin
22 fin
```

La structure du système multi-agent obtenue est illustrée par la figure 5.6. Le système multi-agent résultant de cette agentification est composé de C agents en interaction. Dans

l'exemple donné, la coalition est composée de trois agents. Chaque agent joue simultanément les rôles *Improver*, *Explorer*, *Manager* et *Learner*. Les interactions entre agents résultent des rôles *Manager* et *Learner*. L'interaction entre les instances du rôle *Manager* consiste à échanger les meilleures solutions connues, tandis que l'interaction entre les instances du rôle *Learner* est relative au mimétisme. Le réseau d'interaction est un graphe complet où chaque agent peut interagir avec n'importe quel autre agent. Ainsi, lors de l'envoi d'une nouvelle meilleure solution connue, tous les agents de la coalition (excepté l'émetteur) reçoivent la solution. Il en va de même lors de l'envoi d'une matrice de poids pour l'apprentissage par mimétisme.

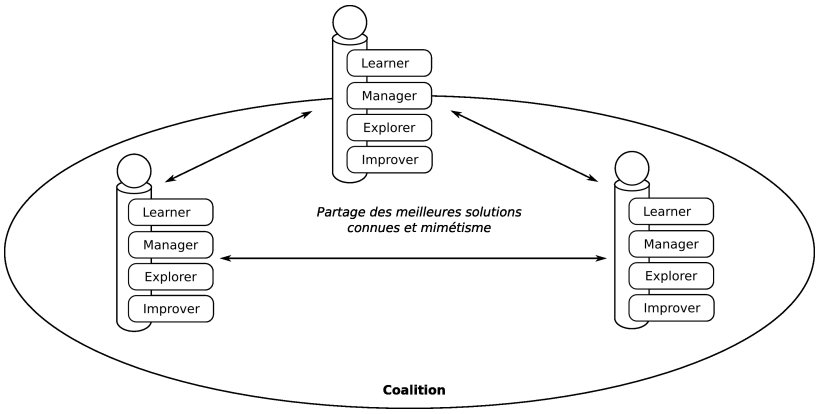

Figure 5.6: Agentification de l'organisation *CBM*

Dans notre approche, la répartition du processus d'optimisation dans la coalition doit permettre d'améliorer les performances globales en s'appuyant sur la coopération entre les agents. L'objectif est d'obtenir un système dans lequel l'exécution d'une coalition de C agents pour une durée T est plus performante que l'exécution d'un unique agent pendant une durée $C * T$. Cette propriété est vérifiée par les expérimentations que nous proposons au chapitre 6.

L'agentification de l'organisation *CBM* que nous venons de décrire, définit un modèle de métaheuristique. Certains éléments de la métaheuristique devront être spécialisés suivant le problème particulier à traiter. En adoptant le principe des approches hyper-heuristiques, nous facilitons cette spécialisation. En effet, les seuls éléments à spécialiser sont les opérateurs d'intensification et de diversification. Les comportements relatifs aux rôles *Manager* et *Learner* peuvent être implantés et réutilisés quelque soit le problème traité. Nous détaillons dans les chapitres suivants cette spécialisation pour deux problèmes d'optimisation.

5.6 Conclusion

Dans ce chapitre nous avons présenté une métaheuristique nommée *CBM* fondée sur la métaphore de la coalition. Dans cette métaheuristique, la recherche de solution est effectuée

par un ensemble d'agents regroupés dans une coalition. Chaque agent est capable d'effectuer indépendamment des autres une recherche à l'aide d'opérateurs de recherche locale et d'adapter sa stratégie suivant un apprentissage par renforcement. La coopération entre agents consiste, d'une part, à partager les meilleures solutions et, d'autre part, à s'échanger des informations sur les stratégies de recherche performantes suivant un comportement de mimétisme.

L'architecture de *CBM* est inspirée des hyper-heuristiques dont le principe est de combiner un ensemble de sous-heuristiques ou d'opérateurs par l'intermédiaire d'une couche stratégique indépendante du problème traité. Cette séparation permet d'obtenir une métaheuristique modulaire et facilite la spécialisation. Cependant, dans *CBM* cette architecture est étendue pour faire apparaître trois couches. La couche inférieure est relative aux opérateurs d'intensification et de diversification, la couche médiane applique une stratégie de recherche par ordonnancement de ces opérateurs, et la couche supérieure adapte la stratégie de recherche à la fois individuellement et collectivement. Cette structure s'appuie sur le modèle organisationnel d'*AMF*.

L'objectif de *CBM* est d'exploiter les différents aspects des systèmes multi-agents pour concevoir une méthode efficace, robuste et modulaire. Ces aspects apparaissent, tout d'abord, dans la distribution de la résolution et dans l'emploi de mécanismes de coopération entre agents, et ensuite, dans l'usage de techniques issues de l'apprentissage artificiel.

La résolution distribuée est mise en œuvre de différentes manières dans *CBM*. En concevant dès le départ *CBM* comme un système décentralisé où chaque agent a une certaine autonomie, et où chacun est interchangeable avec un autre, il devient plus aisé de la paralléliser sur des réseaux d'ordinateurs standards en communication asynchrone et sans mémoire centrale partagée. D'autre part, cette distribution a permis d'envisager différents modes de coopération entre agents que sont le partage de solutions, la mise en concurrence de différentes stratégies de recherche et le mimétisme.

L'efficacité de *CBM* et de ces différents composants sont évalués en traitant un problème d'optimisation de tournées de véhicules dans le chapitre 6 puis un problème de positionnement dans le chapitre 7.

TROISIÈME PARTIE

Application de *CBM* à la résolution de problèmes d'optimisation de réseaux de transports

CHAPITRE 6

APPLICATION DE *CBM* AU PROBLÈME DE TOURNÉES DE VÉHICULES

Sommaire

6.1 Introduction

L'objectif de ce chapitre est de montrer l'application de l'approche CBM sur le problème de tournées de véhicules (*VRP*, *Vehicle Routing Problem*). Ce problème consiste à déterminer un ensemble optimal de circuits afin de desservir un ensemble de clients. À travers cette application, nous souhaitons tout d'abord illustrer la spécialisation dans *CBM* et montrer l'intérêt du choix d'une architecture proche des hyper-heuristiques. Ensuite, l'un des objectifs de cette application au *VRP* est d'évaluer de manière expérimentale l'apport des mécanismes d'apprentissage et de coopération. Enfin, nous souhaitons confronter *CBM* à des approches existantes et comparer les résultats en considérant la qualité des solutions obtenues et le temps d'exécution.

Ce chapitre débute par une présentation du *VRP* et des approches de résolution existantes. Ensuite, la section 6.4 détaille la spécialisation de *CBM* pour traiter le *VRP*. Après une description des benchmarks dans la section 6.5, les résultats expérimentaux de *CBM* sont analysés dans les sections 6.6 et 6.7. Un bilan de ces expérimentations est dressé dans la dernière section du chapitre.

6.2 Le problème de tournées de véhicules

La formulation actuelle du *VRP* a été établie en 1959 dans [Dantzig and Ramser, 1959]. L'application envisagée par les auteurs est la détermination de tournées de camions-citernes pour l'approvisionnement de stations d'essence. L'objectif est de trouver les trajets de moindre coût desservant l'ensemble des stations considérées. Chaque station est associée à un volume demandé et les véhicules ont une capacité qu'ils ne peuvent pas dépasser. Cette variante du *VRP* avec contrainte de capacité est nommée *CVRP* pour *Capacitated Vehicle Routing Problem*. La formulation du *VRP* qui est décrite ci-dessous prend en compte une contrainte de durée en plus de la contrainte de capacité. Cette contrainte supplémentaire impose que la durée de chaque trajet ne doit pas dépasser une valeur spécifiée.

Le *VRP* avec contrainte de capacité et de durée, est défini sur un graphe orienté $G(V, E)$ où $V = \{v_0, ..., v_n\}$ est l'ensemble des nœuds et $E = \{(v_i, v_j)/v_i, v_j \in V; i \neq j\}$ représente l'ensemble des arcs. Dans ce graphe, le nœud v_0 correspond au dépôt et les autres nœuds sont les clients. A chaque client $v_i, i \in \{1, ..., n\}$ est associé une quantité q_i de produits à livrer et un temps d'attente δ_i pour le service. A chaque arc (v_i, v_j) est associé un temps de parcours $c_{i,j}$. Une solution admissible du *VRP* est un ensemble de trajets définis sur le graphe G. Cet ensemble de trajets est soumis aux contraintes suivantes :

- chaque trajet débute et se termine par le dépôt,
- chaque client est desservi par un et un seul véhicule,
- la demande totale de chaque trajet ne doit pas dépasser la capacité des véhicules Q,
- la durée de chaque trajet ne doit pas excéder D.

Afin de décrire les trajets, nous introduisons une variable binaire x_{ijk} qui est égale à 1 si le trajet k emprunte l'arc (v_i, v_j). Cette variable, utilisée dans le modèle de flot à trois indices [Crainic and Semet, 2006], permet ici de simplifier la formulation de la fonction objectif. En utilisant cette variable binaire, il est aisé de donner une formulation du coût d'un trajet. Le coût ou la durée d'un trajet k correspond à la somme des durées des arcs empruntés, additionnée à la somme des temps d'attente des nœuds desservis. Ce coût décrit par la formule 6.1 est noté $C(R_k)$.

$$C(R_k) = \sum_{(v_i,v_j)\in E} c_{ij}x_{ijk} + \sum_{v_i\in V\setminus\{v_0\}} \sum_{v_j\in V} \delta_i x_{ijk} \tag{6.1}$$

Le coût d'une solution du *VRP* est égal à la durée totale des trajets. Ainsi, l'objectif est de minimiser la somme des durées des trajets. Le *VRP* avec contrainte de capacité et de durée s'écrit alors :

$$\text{Minimiser} \sum_{k=1}^{m} C(R_k) \tag{6.2}$$

En respectant les contraintes suivantes :

$$\sum_{k=1}^{m} \sum_{v_j\in V} x_{ijk} = 1 \qquad v_i \in V \setminus \{v_0\} \tag{6.3}$$

$$\sum_{v_i\in V} x_{ilk} = \sum_{v_j\in V} x_{ljk} \qquad v_l \in V \setminus \{v_0\}, 1 \le k \le m \tag{6.4}$$

$$\sum_{v_j\in V\setminus\{v_0\}} x_{0jk} = 1 \qquad 1 \le k \le m \tag{6.5}$$

$$\sum_{v_i\in V\setminus\{v_0\}} x_{i0k} = 1 \qquad 1 \le k \le m \tag{6.6}$$

$$\sum_{v_i\in V\setminus\{v_0\}} \sum_{v_j\in V} q_i x_{ijk} \le Q \qquad 1 \le k \le m \tag{6.7}$$

$$\sum_{v_i\in V\setminus\{v_0\}} \sum_{v_j\in V} \delta_i x_{ijk} + \sum_{(v_i,v_j)\in E} c_{ij}x_{ijk} \le D \qquad 1 \le k \le m \tag{6.8}$$

$$\sum_{v_i\in S, v_j\notin S} x_{ijk} \ge \sum_{v_i\in V} x_{ljk} \qquad S \subset V \setminus \{v_0\}, l \in S, 1 \le k \le m \tag{6.9}$$

$$x_{ijk} \in \{0,1\} \qquad (v_i, v_j) \in E, 1 \le k \le m \tag{6.10}$$

Dans cette formulation, les contraintes 6.3 imposent que chaque client soit desservi une et une seule fois. Les contraintes 6.4 garantissent la continuité des trajets. Les contraintes 6.5 et 6.6 assurent que chaque trajet démarre et s'achève au dépôt. Les contraintes 6.7 et 6.8 sont les contraintes de capacité et de durée et les contraintes 6.9 évitent le passage d'un véhicule par le dépôt en dehors du départ et de l'arrivée. Enfin, les contraintes 6.10 sont les contraintes de binarité.

Il est possible à partir de cette formulation du *VRP* de traiter les instances avec contrainte de capacité uniquement. Pour cela, il suffit de considérer que le temps d'attente δ_i est égal à 0 pour chaque nœud, et que la durée maximale D d'un trajet est une valeur suffisamment grande. Ainsi, les instances de *VRP* traitées par la suite sont, soit des instances avec la contrainte de capacité uniquement, soit des instances avec les contraintes de capacité et de durée.

6.3 Approches de résolution existantes

Le *VRP* présenté dans la section précédente est un problème difficile à résoudre de façon optimale. Les différentes variantes avec et sans contrainte de durée sont des problèmes $\mathcal{NP}$-durs. L'étude de la complexité du *VRP* a été réalisée par Lenstra et Rinnooy Kan dans [Lenstra and Rinnooy Kan, 1981]. Ainsi, le *VRP* ne peut être que rarement résolu de manière exacte pour des instances de plus de 100 clients. Les approches de résolution exactes performantes pour le *VRP* sont principalement les algorithmes de séparations et évaluations progressives (*Branch and bound*) et les algorithmes de séparations et coupes (*Branch and cut*) [Crainic and Semet, 2006].

Un algorithme de séparations et évaluations progressives consiste globalement à construire un arbre de recherche en séparant progressivement des sous-ensembles de l'espace de recherche. Les sous-ensembles sont ensuite évalués afin de déterminer si l'exploration doit ou non continuer sur ceux-ci. Une revue des différentes méthodes de séparations et évaluations appliquées au *VRP* est réalisée dans [Toth and Vigo, 2001]. La méthode de séparations et coupes est une amélioration de la méthode de séparations et évaluations progressives. Elle consiste à génèrer un ensemble de coupes avant de procéder par séparations et évaluations progressives. Une présentation des méthodes de séparations et coupes appliquées au *VRP* est donnée dans [Naddef and Rinaldi, 2001].

Dans [Crainic and Semet, 2006], les auteurs observent que ces méthodes exactes pour la résolution du *VRP* ne permettent pas de traiter des instances de plus de 100 clients en un temps raisonnable. Cette limitation justifie l'emploi de méthodes heuristiques telles que l'approche *CBM* [Meignan et al., 2008a].

Nous limitons notre présentation des heuristiques appliquées au *VRP* à quatre approches parmi les plus performantes : *GTS* [Toth and Vigo, 2003], *UTSA* [Cordeau et al., 2004], *AGES* [Mester and Bräysy, 2005, 2007] et un algorithme mémétique [Prins, 2004]. Ces différentes approches servent de référence de base permettant la comparaison des résultats obtenus avec *CBM*.

GTS (*Granular Tabu Search*) [Toth and Vigo, 2003] et *UTSA* (*Unified Tabu Search Algorithm*) [Cordeau et al., 2004] sont deux approches fondées sur la recherche tabou. Dans *GTS*, le voisinage utilisé est fondé sur le déplacement d'un arc entre les trajets et à l'intérieur d'un trajet. La principale originalité de l'approche est de restreindre le voisinage

d'une solution en y supprimant les solutions contenant des arcs ayant un coût élevé. Le paramètre fixant ce coût est ajusté dynamiquement afin de favoriser l'intensification ou la diversification de la recherche. *UTSA* est une recherche tabou utilisant un voisinage *1-move* réduit. Dans ce voisinage, l'insertion d'un client dans un autre trajet ne s'effectue que sur l'arc le plus proche du client. *UTSA* intègre une procédure de diversification qui consiste à considérer le dépôt comme étant l'un des autres nœuds du graphe pendant un nombre limité d'itérations.

AGES (*Active Guided Evolutionary Strategies*) [Mester and Bräysy, 2005, 2007] et l'algorithme mémétique [Prins, 2004] sont deux approches à base de population. *AGES* est une approche combinant une recherche locale guidée et un algorithme évolutionniste dans une procédure itérative à deux étapes. La première est une recherche locale guidée utilisant 3 ou 5 heuristiques différentes et appliquant des pénalités sur les arcs du graphe de l'instance de *VRP*. La seconde étape est un algorithme évolutionniste de type *(1+1)-evolution strategy* faisant évoluer les solutions individuellement en appliquant un opérateur de mutation. L'algorithme mémétique proposé dans [Prins, 2004] utilise un opérateur de croisement et une procédure de recherche locale pour faire évoluer une population de solution. La procédure de croisement utilisée est un croisement ordonné (*Order Crossover*, *OX*) s'appliquant sur un vecteur de nœuds dans lequel le dépôt n'est pas présent. La procédure de recherche locale est fondée sur les voisinages *2-opt*, *1-move*, *1-swap*, *edge-swap* et *edge-move* présentés dans la section suivante. L'algorithme global de recherche intègre un "restart" avec une modification du taux d'application de la recherche locale.

Une présentation plus détaillée des heuristiques les plus performantes pour le *VRP* est réalisée dans [Cordeau et al., 2002].

6.4 Spécialisation de *CBM* pour la résolution du *VRP*

Dans cette section, nous présentons le processus de spécialisation de *CBM* pour résoudre le *VRP*. Nous rappelons d'abord l'objectif de cette étape de spécialisation puis nous l'appliquons sur notre approche. A l'issue de cette spécialisation, tous les éléments de *CBM* seront entièrement définis pour résoudre le *VRP*.

6.4.1 Principe de la spécialisation de *CBM*

La spécialisation correspond à la troisième étape décrite dans les règles méthodologiques de l'approche *AMF*. Elle consiste à passer d'une métaheuristique, qui est un modèle générique de résolution, à une méthode effectivement capable de résoudre un problème donné. Pour cela, la connaissance spécifique au problème traité doit être introduite, notamment en spécifiant les opérateurs de bas niveau.

Dans le chapitre précédent *CBM* a été présentée comme un système multi-agent. Ce

système permet de résoudre un problème d'optimisation à partir d'un ensemble d'agents regroupés dans une coalition. Ces agents coopèrent pour rechercher la meilleure solution à l'instance de problème qui leur a été soumise. Dans cette coalition, chaque agent fait évoluer une solution à l'aide d'un ensemble d'opérateurs d'intensification et de diversification ordonnancé par un processus de décision. Le modèle de résolution proposé est potentiellement applicable à plusieurs problèmes d'optimisation, et il est nécessaire de le spécialiser pour établir une méthode effectivement capable de résoudre un problème particulier.

L'architecture de résolution adoptée dans *CBM* est proche des hyper-heuristiques, ce qui limite le nombre d'éléments spécifiques au problème. Les seuls composants à définir sont les opérateurs d'intensification et de diversification, comme l'illustre la figure 6.1. Le processus de sélection d'opérateur et les mécanismes d'apprentissage sont indépendants du problème et ne nécessitent pas de spécialisation.

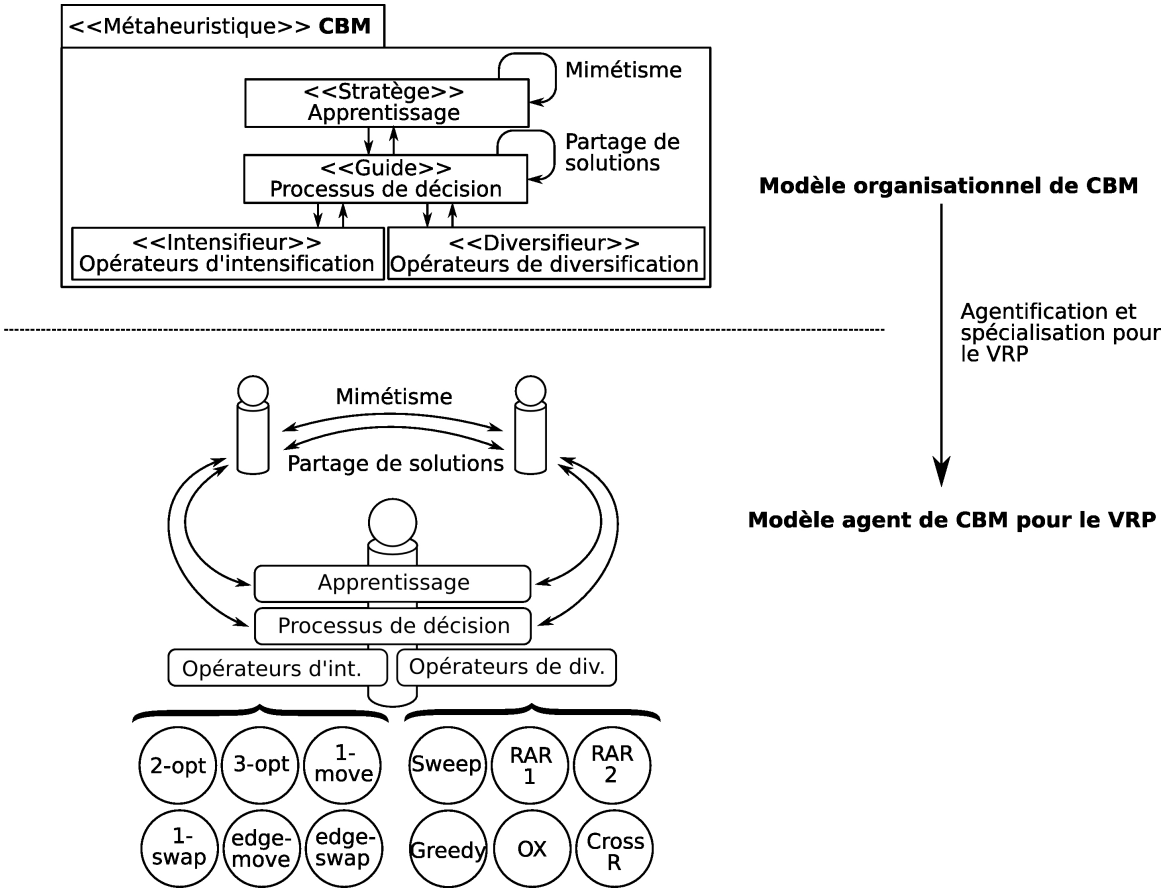

Figure 6.1: Principe de spécialisation de *CBM* pour le *VRP*

Pour résoudre le *VRP*, nous introduisons six opérateurs de diversification et six opérateurs d'intensification. Les opérateurs d'intensification sont des procédures de recherche locale et les opérateurs de diversification sont des procédures de génération, mutation et croisement de solutions.

6.4.2 Opérateurs d'intensification pour le *VRP*

Les opérateurs d'intensification utilisés dans *CBM* permettent d'obtenir un minimum local à partir d'une solution initiale. Ils utilisent une structure de voisinage afin de faire évoluer la solution par mouvements successifs dans le voisinage. Les procédures de recherche locale utilisées permettent uniquement les déplacements qui améliorent la fitness. La recherche locale se termine lorsque la solution est un minimum local, c'est-à-dire, qu'aucun déplacement dans le voisinage de la solution ne permet d'améliorer la fitness. Ce type de recherche est nommé descente locale.

Afin d'illustrer la notion de voisinage dans le cadre du *VRP*, nous considérons qu'une solution d'une instance de *VRP* est décrite par un ensemble de trajets, c'est-à-dire plusieurs sous-ensembles de nœuds et d'arcs définis sur un graphe. Une structure de voisinage permet de définir pour chaque solution, un ensemble de solutions voisines. La plupart des méthodes de recherche locale utilisées pour résoudre le *VRP* utilisent des permutations ou des déplacements de nœuds ou d'arcs pour définir le voisinage d'une solution. Par exemple, il est possible de définir le voisinage d'une solution de *VRP* comme étant constitué de l'ensemble des solutions obtenues en permutant deux nœuds de deux trajets différentes. Ce voisinage nommé *1-swap* est détaillé dans la suite.

Lorsqu'une structure de voisinage a été définie, il existe plusieurs variantes possibles pour réaliser une descente locale. Ces variantes se distinguent par le critère de choix de la solution voisine à chaque itération. Par exemple, il est possible de choisir la solution maximisant la fitness parmi le voisinage ou encore de choisir la première solution rencontrée améliorant la fitness. Généralement, les procédures de recherche locale utilisées pour résoudre le *VRP* adoptent cette seconde alternative qui évite d'évaluer exhaustivement l'ensemble des voisins d'une solution.

L'algorithme 5 présente la procédure de recherche locale réalisée par les opérateurs d'intensification dans *CBM*. Cet algorithme permet à partir d'une solution s d'arriver à un minimum local sur le voisinage $\mathcal{N}$. Les fonctions *ExisteSuivant* et *Suivant* permettent une énumération des solutions du voisinage, et la fonction "*f*" permet d'obtenir le coût d'une solution. Dans cet algorithme de descente locale, la solution courante est remplacée par la solution voisine dès que la fitness est améliorée. Ces déplacements s'exécutent tant qu'un minimum local n'est pas atteint. La variable booléenne *déplacement* permet d'identifier ce cas.

Pour traiter le *VRP*, nous proposons d'utiliser six structures de voisinage : *2-opt*, *3-opt*, *1-swap*, *1-move*, *edge-swap* et *edge-move*. Ces structures de voisinage, couramment utilisées pour résoudre le *VRP* [Laporte et al., 2000], ont permis de définir six opérateurs d'intensification.

Algorithme 5 : Procédure de descente locale

```
répéter
    déplacement ← Faux
    tant que ExisteSuivant(N(s)) et déplacement = Faux faire
        v ← Suivant(N(s))
        si f(v) < f(s) alors
            s ← v
            déplacement ← Vrai
        fin
    fin
jusqu'à déplacement = Faux
```

Voisinages *2-opt* et *3-opt*

Les voisinages *2-opt* et *3-opt* sont des cas particuliers du voisinage λ-*opt* [Lin, 1965] initialement utilisé pour la résolution du TSP. Les deux procédures de recherche locale utilisant ces voisinages opèrent sur les trajets d'une solution pris séparément. Ainsi, les nœuds composant une solution ne sont pas échangés entre les trajets.

Un déplacement dans le voisinage λ-*opt* d'une solution consiste à supprimer λ arcs d'un trajet et à reconnecter d'une autre manière les segments restants. La figure 6.2 présente un exemple de voisinage *2-opt* d'une solution. Dans cette figure, une instance de *VRP* constituée de 7 clients et d'un dépôt est considérée. Les solutions S_1 à S_9 sont les solutions voisines de S_0. Chacune de ces neuf solutions est obtenue en supprimant deux arcs d'un même trajet à la solution S_0 et en reconnectant les segments restants d'une autre manière. La fitness de chaque solution est indiquée entre parenthèses. Sur cet exemple, si une procédure de descente locale est appliquée à partir de la solution S_0, le premier mouvement effectué dans le voisinage *2-opt* de la solution correspondra à la solution S_7. En effet, il s'agit de la seule solution du voisinage de S_0 améliorant la fitness.

La complexité en temps, du parcours du voisinage λ-*opt* d'une solution est $O(n^\lambda)$, où n représente la taille du problème (nombre de clients ou de nœuds). Dans *CBM* λ ne dépasse pas 3 pour éviter une durée trop importante de l'application des opérateurs.

Voisinages *1-move*, *edge-move*, *1-swap* et *edge-swap*

Les voisinages *1-move*, *edge-move*, *1-swap* et *edge-swap* sont aussi appelés respectivement *1-string relocation*, *2-string relocation*, *1-string exchange* et *2-string exchange* [Laporte et al., 2000]. Il s'agit de voisinages complémentaires aux *2-opt* et *3-opt*. Ils consistent à modifier deux trajets lors d'un déplacement d'arc ou de sommet. Les voisinages *1-move* et *edge-move* d'une solution sont obtenus en insérant respectivement un nœud ou un arc d'un trajet dans un autre. Les *1-swap* et *edge-swap* d'une solution sont obtenus en permutant respectivement un nœud ou un arc entre deux trajets.

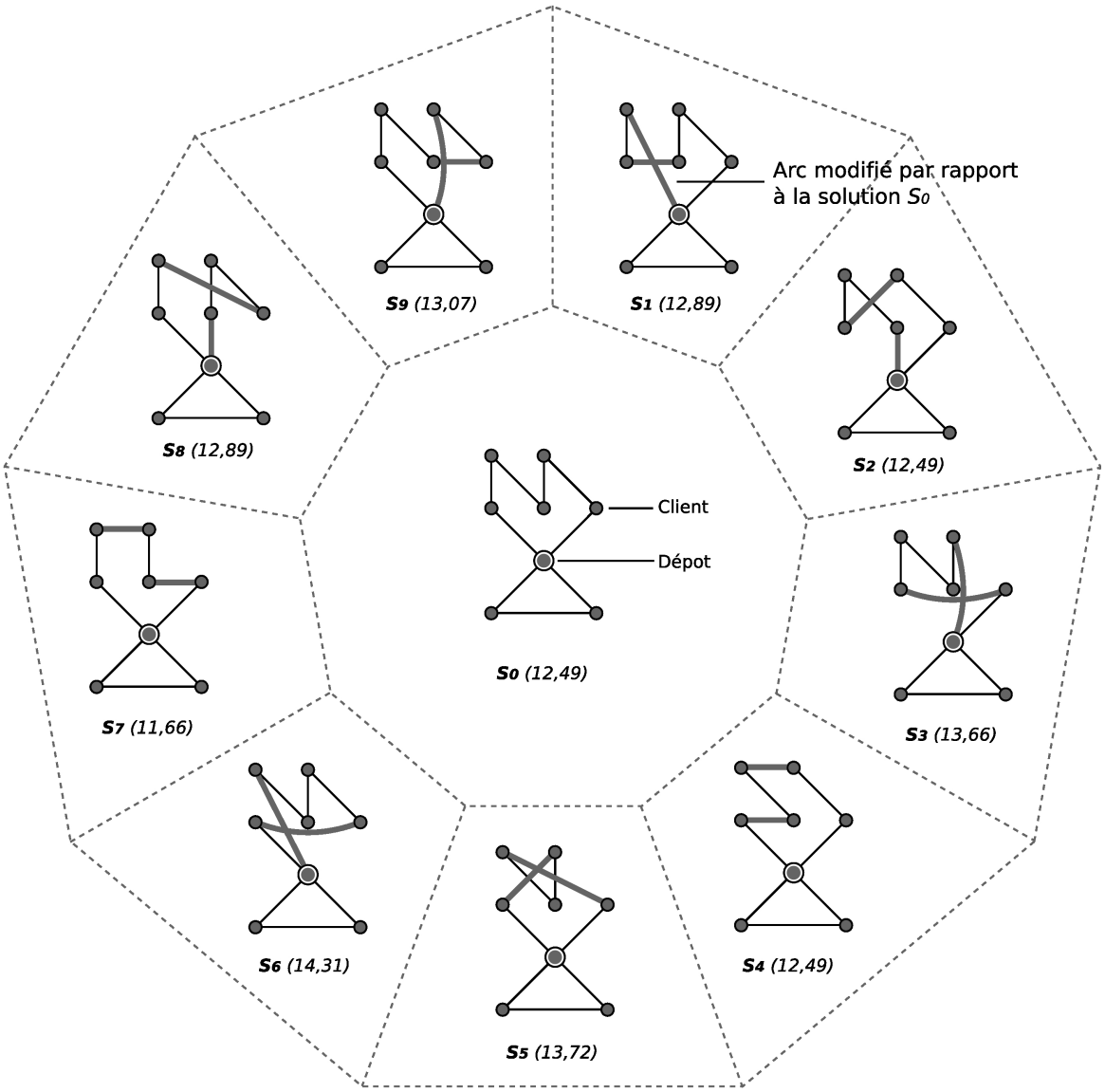

Figure 6.2: Exemple de voisinage *2-opt* d'une solution

Lors du parcours du voisinage d'une solution en utilisant ces structures simples il est nécessaire de vérifier que les solutions respectent les contraintes de capacité et de durée. Une solution ne respectant pas ces contraintes n'est pas considérée dans le voisinage. Cette vérification des contraintes est cependant inutile pour les voisinages *2-opt* et *3-opt* puisque lors du choix d'une solution voisine (améliorant la fitness), la durée des trajets ne peut que diminuer et la capacité de chaque trajet reste inchangée.

6.4.3 Opérateurs de diversification pour le *VRP*

Les opérateurs de diversification dans *CBM* sont des procédures de génération, de mutation ou de croisement. Les opérateurs de génération permettent de construire une nouvelle solution. Les opérateurs de mutation appliquent une perturbation aléatoire à une solution

initiale. Enfin, les opérateurs de croisement combinent les composantes de deux solutions afin de produire une nouvelle solution. Pour traiter le *VRP*, six opérateurs de diversification ont été utilisés :

- deux opérateurs de génération nommés génération gloutonne et génération par rotation,
- deux opérateurs de mutation correspondant à une procédure de suppression/insertion paramétrée avec deux taux de perturbation différents,
- deux opérateurs de croisement intitulés croisement par insertion et croisement ordonné.

Nous détaillons ces opérateurs ci-après.

La génération gloutonne est une procédure simple qui consiste à générer une solution en insérant progressivement les nœuds non desservis dans une solution partielle. Pour cela, chaque nœud est sélectionné dans un ordre aléatoire et est inséré dans la solution partielle de manière à avoir le coût minimum tout en respectant les contraintes de capacités et de durée.

L'algorithme de génération par rotation a été introduit dans [Wren and Holliday, 1972] sous le nom de *sweep algorithm*. Cet algorithme ne peut s'appliquer que pour des instances planes du *VRP*, c'est-à-dire, pour lesquelles les nœuds correspondent à des points d'un plan. La *génération par rotation* consiste à construire de manière séquentielle chaque trajet à partir d'une liste ordonnée de nœuds. L'ordre d'insertion des nœuds est obtenu en faisant tourner un rayon centré sur le dépôt (nœud v_0). Chaque nœud est inséré dans le trajet courant tant que la capacité et la durée maximale ne sont pas atteintes. Si le nœud ne peut pas être inséré dans le trajet alors un nouveau trajet est créée.

Les deux opérateurs de mutation sont basés sur une même procédure de suppression/insertion paramétrée avec des taux de suppression différents, obtenus suite à plusieurs expérimentations. La procédure de suppression/insertion consiste à supprimer un ensemble de nœuds des trajets, puis à les réinsérer à une position aléatoire tout en respectant les contraintes de durée et de capacité. Cette procédure peut être vue comme une suite de déplacements aléatoires dans un voisinage de type *1-move* de la solution. Nous précisons que le voisinage est plus étendu que celui de la version classique *1-move* utilisée dans l'intensification, car un nœud peut être réinséré dans son trajet initial.

Dans le croisement par insertion, une nouvelle solution est obtenue en insérant une trajet d'une première solution dans une seconde solution. Pour obtenir une solution valide, chaque nœud contenu dans le trajet inséré est supprimé des trajets de la seconde solution. Suite à cette insertion, une procédure cherchant à regrouper les trajets est appliquée afin de ne pas avoir une augmentation trop importante de leur nombre.

Le croisement ordonné (*Order Crossover*, *OX*) [Oliver et al., 1987] est un croisement à deux points où l'ordre relatif des nœuds tend à être conservé. Pour cet algorithme de croisement, chacune des deux solutions initiales est mise sous la forme d'un vecteur de

nœuds correspondant à la concaténation des trajets de la solution sans les passages au dépôt v_0. Ensuite, deux points de coupe sont choisis aléatoirement dans la première solution. La séquence de nœuds entre les deux points de coupe est complétée en ajoutant les nœuds de la seconde solution pris dans l'ordre et en supprimant les nœuds déjà présents. Suite à cette combinaison, le vecteur de nœuds résultant est découpé afin d'obtenir des trajets respectant les contraintes de capacité et de durée.

6.5 Présentation des benchmarks

Pour évaluer l'approche *CBM*, les deux benchmarks proposés dans [Christofides et al., 1979] et [Golden et al., 1998] ont été utilisés. Le premier est sûrement l'un des plus utilisés pour le *VRP*. Le second benchmark fournit des instances de *VRP* de plus grande taille. Les instances composant ces deux benchmarks n'ont pas été résolues nécessairement de manière exacte. Néanmoins, les meilleurs résultats récemment obtenus constituent une bonne approximation des valeurs optimales. Nous utilisons pour cela les résultats présentés dans [Mester and Bräysy, 2007].

Les caractéristiques des deux benchmarks sont résumées dans les tableaux 6.1 et 6.2. Dans ces tableaux sont indiqués dans les trois premières colonnes le numéro de l'instance, sa taille (nombre de clients) et le type de contraintes. Dans la colonne *Contraintes*, un *C* indique une contrainte de capacité et un *D* indique une contrainte de durée. L'avant dernière colonne indique le type de distribution des clients et la dernière colonne reporte la valeur de la meilleure solution connue tirée de [Mester and Bräysy, 2007].

Le benchmark Christofides et al. est composé de 14 instances de 50 à 199 clients. La moitié des instances ont uniquement une contrainte de capacité, et l'autre moitié ont une contrainte de durée en plus. Il s'agit d'instances euclidiennes, ce qui implique que la matrice des distances entre les nœuds est symétrique et que les valeurs de coût sont des distances euclidiennes. Dans les dix premières instances, la position des clients a été générée à partir d'une distribution uniforme aléatoire. Pour les quatre dernières instances, la distribution des clients présente des clusters, c'est-à-dire des zones de concentration variable de clients. Ces problèmes reflètent ainsi des cas réels d'application pratique.

Le benchmark Golden et al., décrit dans le tableau 6.2, est composé de 20 instances de 200 à 483 clients. Les huit premières instances ont une contrainte de capacité seulement et les autres ont en plus une contrainte de durée. Ces 20 instances sont euclidiennes, comme pour le benchmark [Christofides et al., 1979] mais la disposition des clients correspond à des structures géométriques régulières. Dans 8 instances, les clients forment des cercles concentriques autour du dépôt. Pour 8 autres instances, la disposition des clients fait apparaître des carrés concentriques avec le dépôt placé soit au centre soit sur un des coins du carré. Enfin, dans les quatre dernières instances, les clients sont disposés sous forme d'une étoile à six branches.

Ces deux benchmarks sont intéressants à plus d'un titre pour évaluer l'approche *CBM*. D'une part, ils regroupent des instances avec une large gamme de tailles, ayant différentes contraintes et différents types de distribution des clients. D'autre part, ces benchmarks sont largement utilisés dans la littérature pour comparer les performances des heuristiques et métaheuristiques quant à la résolution du *VRP*.

Instance	Taille	Contraintes	Distribution	Meilleure connue
1	50	C	Aléatoire uniforme	524,61
2	75	C	Aléatoire uniforme	835,26
3	100	C	Aléatoire uniforme	826,14
4	150	C	Aléatoire uniforme	1028,42
5	199	C	Aléatoire uniforme	1291,29
6	50	C, D	Aléatoire uniforme	555,43
7	75	C, D	Aléatoire uniforme	909,68
8	100	C, D	Aléatoire uniforme	865,94
9	150	C, D	Aléatoire uniforme	1162,55
10	199	C, D	Aléatoire uniforme	1395,85
11	120	C	Avec concentration	1042,11
12	100	C	Avec concentration	819,56
13	120	C, D	Avec concentration	1541,14
14	100	C, D	Avec concentration	866,37

Table 6.1: Caractéristiques des instances du benchmark Christofides et al.

6.6 Analyse expérimentale des composants de *CBM*

Cette section propose une validation expérimentale des principaux composants de *CBM*. Tout d'abord, le processus de décision est testé afin de valider la stratégie d'alternance proposée entre une phase de diversification et une phase d'intensification. Ensuite, sont présentés des résultats expérimentaux sur l'apport des mécanismes d'apprentissages. Nous essayons notamment de mettre en évidence l'apport de l'apprentissage réalisé par l'agent seul et ensuite celui réalisé par la mise en œuvre du mimétisme. Enfin, il s'agit d'illustrer l'intérêt de la coopération entre les agents en faisant varier leur nombre à durée d'exécution constante.

6.6.1 Stratégie d'alternance entre la diversification et l'intensification

Le premier composant que nous proposons de tester pour valider l'approche *CBM* est le processus de décision et tout particulièrement son initialisation. Dans *CBM*, chaque agent dispose d'un processus de décision qui lui permet de choisir l'opérateur à appliquer. Il est fondé sur une matrice condition/opérateur décrite dans la section 5.4.1. Les conditions se

Instance	Taille	Contraintes	Distribution	Meilleure connue
1	240	C	Cercles concentriques	5627,54
2	320	C	Cercles concentriques	8447,92
3	400	C	Cercles concentriques	11036,22
4	480	C	Cercles concentriques	1324,52
5	200	C	Cercles concentriques	6460,98
6	280	C	Cercles concentriques	8412,80
7	360	C	Cercles concentriques	10195,56
8	440	C	Cercles concentriques	11663,55
9	255	C, D	Carrés concentriques	583,39
10	323	C, D	Carrés concentriques	741.56
11	399	C, D	Carrés concentriques	918,45
12	483	C, D	Carrés concentriques	1107,19
13	252	C, D	Carrés concentriques	859,11
14	320	C, D	Carrés concentriques	1081,31
15	396	C, D	Carrés concentriques	1345,23
16	480	C, D	Carrés concentriques	1622,69
17	240	C, D	Étoile à six branches	707,79
18	300	C, D	Étoile à six branches	998,73
19	360	C, D	Étoile à six branches	1366,86
20	420	C, D	Étoile à six branches	1820,09

Table 6.2: Caractéristiques des instances du benchmark Golden et al.

rapportent à l'opérateur appliqué précédemment, et chaque valeur de la matrice correspond au potentiel de choix d'un opérateur pour une condition donnée. Lors de l'initialisation de la matrice de décision, plusieurs valeurs de couples condition/opérateur sont fixées à 0 afin d'alterner correctement l'application d'opérateurs d'intensification et d'opérateurs de diversification. Le fait d'initialiser la valeur de poids d'un couple condition/opérateur à 0 permet de rendre nulle la probabilité de sélection de l'opérateur dans la condition donnée.

Les règles d'initialisation de la matrice de décision retenues dans le cadre de *CBM* impliquent d'alterner une étape de diversification avec une phase longue d'intensification. La diversification consiste à appliquer ponctuellement un opérateur de diversification tandis que la phase d'intensification correspond à l'application d'un ensemble d'opérateurs d'intensification jusqu'à l'obtention d'un minimum local sur l'ensemble des structures de voisinage considérées. Cependant, le choix de cette stratégie spécifique d'alternance a été établi à partir d'une comparaison des performances avec d'autres stratégies possibles et envisageables d'ordonnancement.

Dans le cadre des hyper-heuristiques Özcan et al. ont analysé différentes stratégies d'alternance entre la diversification et l'intensification [Özcan et al., 2008]. Parmi les stratégies proposées, nous en avons retenu trois pour le *VRP* : (i) choix effectué sans distinction entre opérateur de diversification et opérateur d'intensification, (ii) alternance entre le choix d'un

opérateur de diversification et le choix d'un opérateur d'intensification, (iii) alternance entre le choix d'un opérateur de diversification avec une phase d'intensification longue se terminant lors de l'obtention d'un minimum local sur l'ensemble des structures de voisinage considérées. Ces trois stratégies sont illustrées dans la figure 6.3. Pour *CBM*, elles se traduisent par des initialisations particulières de la matrice de décision.

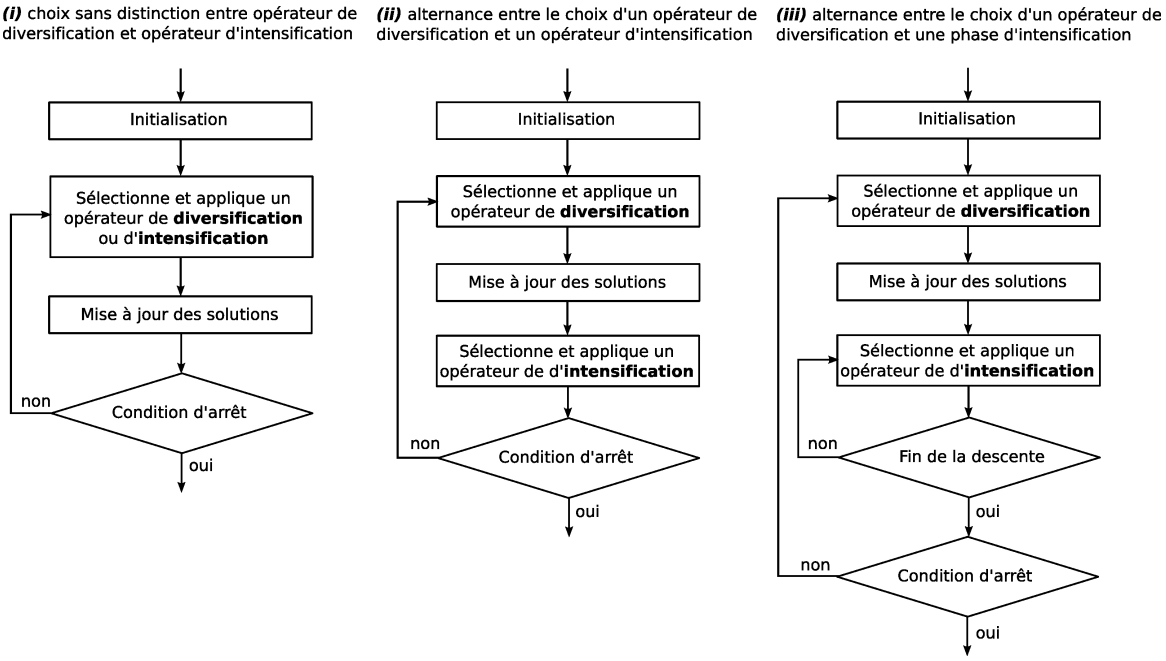

Figure 6.3: Stratégies d'alternance entre opérateurs de diversification et opérateurs d'intensification, inspiré de [Özcan et al., 2008]

Voici comment les initialisations particulières de la matrice de décision sont réalisées pour chacune des stratégies de sélection d'opérateur :

- Pour obtenir un choix d'opérateur sans distinction entre la diversification et l'intensification, l'ensemble des valeurs des couples condition/opérateur de la matrice de décision sont initialisées avec une valeur identique $\alpha > 0$. Ainsi, quelle que soit la condition, chaque opérateur aura la même probabilité de sélection.
- Pour alterner le choix entre un opérateur de diversification et un opérateur d'intensification, nous avons initialisé la matrice de sorte que la probabilité de choix d'un opérateur de diversification soit nulle après l'application d'un opérateur de diversification, et que la probabilité de choix d'un opérateur d'intensification soit nulle après l'application d'un opérateur d'intensification. Ainsi, pour la condition où la solution courante n'est pas un optimum local, la valeur de poids des opérateurs d'intensification est fixée à $\alpha > 0$ et le poids des opérateurs de diversification est fixé à 0. Pour l'ensemble des autres conditions, la valeur de poids des opérateurs d'intensification est fixée à 0 et le poids des opérateurs de diversification est fixé à $\alpha > 0$.
- L'alternance entre le choix d'un opérateur de diversification et une phase d'intensification est obtenue en distinguant le cas où la solution courante est un optimum

local sur l'ensemble des structures de voisinage considérées et les autres cas. Pour la condition où la solution courante est un optimum local sur l'ensemble des structures de voisinage considérées, le poids des opérateurs de diversification est fixé à $\alpha > 0$ et le poids des opérateurs d'intensification est fixé à 0. Pour les autres conditions, la valeur de poids des opérateurs d'intensification est fixée à $\alpha > 0$ et le poids des opérateurs de diversification est fixé à 0.

Afin d'éviter l'application inutile d'un opérateur d'intensification utilisant un voisinage $\mathcal{N}_i$ lorsque la solution est un optimum local sur $\mathcal{N}_i$, les couples condition/opérateur de la matrice de décision correspondant sont fixés à 0.

Pour tester ces trois stratégies, *CBM* a été implanté avec un seul agent et sans le rôle *Stratège*, c'est-à-dire sans apprentissage. Ainsi, les valeurs de la matrice de décision ne sont pas modifiées lors de l'optimisation et l'agent conserve la même stratégie de choix des opérateurs. Les trois stratégies ont été testées sur le benchmark Christofides et al. avec les 12 opérateurs précédemment détaillés. Pour chaque stratégie et chaque instance, *CBM* a été exécutée 20 fois sur une durée de 1 minute.

Les résultats moyens obtenus sur 20 exécutions de ces trois stratégies sont illustrés dans le diagramme 6.4 pour chaque jeu de test. La série *Choix aléatoire* correspond à la stratégie qui consiste à choisir indifféremment un opérateur de diversification ou d'intensification. La série *Choix alterné* correspond à la stratégie d'alternance entre opérateurs de diversification et opérateurs d'intensification, et la série nommée *Phase d'intensification* correspond à la stratégie qui alterne entre le choix d'un opérateur de diversification et une phase d'intensification. Pour chaque instance, l'écart moyen à la meilleure solution connue de l'instance est représenté sur le diagramme 6.4.

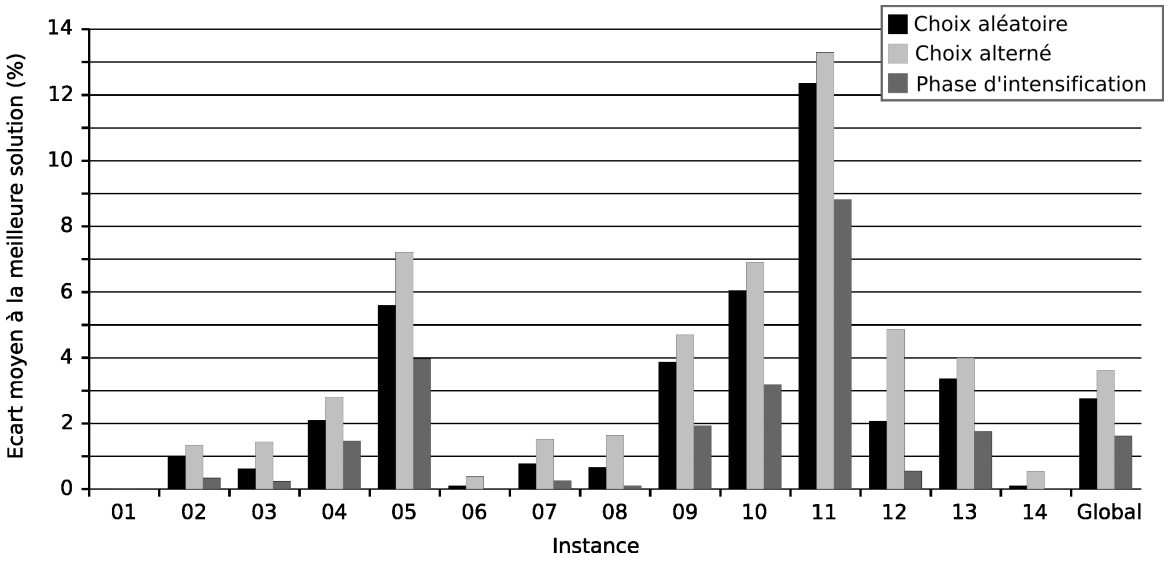

Figure 6.4: Résultats comparés des stratégies d'alternance entre opérateurs de diversification et opérateurs d'intensification, réalisés sur les instances Christofides et al.

À partir de ces résultats, nous remarquons sur l'ensemble des instances, que la stratégie

de choix alterné est la moins performante et que la stratégie avec phase d'intensification est très nettement la plus performante. Le résultat moyen de ces deux stratégies est respectivement 3,6% et 1,6% d'écart moyen à la meilleure solution connue. L'ensemble de ces résultats laisse supposer que l'application trop fréquente d'un opérateur de diversification ne permet pas d'obtenir de bons résultats. Cette observation confirme en quelque sorte le bien-fondé d'une alternance entre le choix d'un opérateur de diversification suivi d'une phase d'intensification se terminant lorsque la solution courante est un minimum local sur l'ensemble des structures de voisinage considérées. Cette stratégie de cycle diversification/intensification fournit également la base temporelle pour l'observation et la mise à jour des poids par apprentissage.

Nous montrons dans les sections suivantes que cette stratégie initiale de choix des opérateurs peut être améliorée par l'apprentissage. Dans cette stratégie initiale, le choix d'un opérateur de diversification ou d'intensification s'effectue de manière équiprobable (poids non nuls et égaux). L'apprentissage va permettre de modifier les poids de la matrice de décision pendant l'optimisation et ainsi privilégier certains opérateurs ou séquences d'opérateurs suivant leur degré d'efficacité supposé et observé.

6.6.2 Mécanismes d'apprentissage

Dans cette section, nous souhaitons valider expérimentalement les deux principaux mécanismes d'apprentissage mis en jeux dans *CBM*. Ces mécanismes d'apprentissage ont pour objectif d'améliorer la stratégie de choix d'opérateurs des agents. Cette stratégie est fondée sur la matrice condition/opérateur propre à chaque agent qui permet de sélectionner les opérateurs à appliquer sur sa solution courante. L'apprentissage consiste à modifier les valeurs de poids de la matrice de décision afin de favoriser le choix d'opérateurs efficaces. Deux types d'apprentissage complémentaires sont employés par les agents : par renforcement et par mimétisme. Ces mécanismes ont été détaillés dans la section 5.4.2. L'apprentissage par renforcement consiste à augmenter le poids des couples condition/opérateur ayant permis d'obtenir une nouvelle meilleure solution. Le mimétisme consiste à imiter le comportement des agents ayant trouvé une bonne solution au niveau de la coalition. Dans ce cas, la matrice de décision de l'agent qui mime est modifiée suivant une moyenne pondérée de sa matrice et de celle de l'agent imité.

Pour observer l'effet produit par les mécanismes d'apprentissage, plusieurs configurations de *CBM* ont été testées. Ces configurations se différencient par l'utilisation ou non des mécanismes d'apprentissage. Trois configurations ont été comparées, (i) sans apprentissage, (ii) avec uniquement l'apprentissage par renforcement, et (iii) avec apprentissage par renforcement et mimétisme. Pour la première configuration, *CBM* a été implantée sans le rôle *Stratège*. La deuxième configuration correspond à une implantation avec le rôle *Stratège* mais sans les interactions de mimétisme entre les différentes instances de ce rôle. La dernière implantation correspond à une version complète de *CBM* telle qu'elle a été décrite

dans la figure 6.1.

Les simulations ont été effectuées sur le benchmark Christofides et al. avec une taille de coalition de 10 agents et un nombre total d'application d'opérateurs limité à 1000 par agent. Pour chaque instance et chaque configuration, *CBM* a été exécutée 50 fois.

Le tableau 6.3 présente les résultats de *CBM* avec et sans apprentissage. Dans ce tableau, les premières colonnes rappellent les caractéristiques des instances. Les colonnes suivantes présentent pour chacune des configurations l'écart moyen à la meilleure solution connue de l'instance, l'intervalle de confiance (IC) à 95% et le temps d'exécution moyen. Les trois dernières lignes du tableau indiquent la moyenne de ces valeurs, respectivement pour les instances avec contrainte de capacité uniquement, pour les instances avec contrainte de durée, et sur l'ensemble des instances. On remarque qu'avec des durées d'exécution quasi similaires, la configuration avec les deux mécanismes d'apprentissage est plus performante que les deux autres configurations. De même, la version avec renforcement améliore la version sans apprentissage. Globalement, l'ajout de capacités d'apprentissage a une influence sur la qualité des résultats. De même, ces observations semblent être vérifiées sur chacune des instances du benchmark prise séparément.

Table 6.3: Comparaison de *CBM* avec et sans apprentissage sur les instances Christofides et al.

Instance				Sans apprentissage			Avec renforcement			Avec renforcement et mimétisme		
	Contrainte	Taille	Meilleure connue	Moyenne	IC (95 %)	Durée (min)	Moyenne	IC (95 %)	Durée (min)	Moyenne	IC (95 %)	Durée (min)
01	C	50	524,61	0,00 %	± 0,00 %	0,04	0,00 %	± 0,00 %	0,04	0,00 %	± 0,00 %	0,04
02	C	75	835,26	1,05 %	± 0,19 %	0,10	1,00 %	± 0,20 %	0,09	0,91 %	± 0,21 %	0,09
03	C	100	826,14	0,52 %	± 0,07 %	0,25	0,43 %	± 0,05 %	0,25	0,40 %	± 0,07 %	0,24
04	C	150	1028,42	1,68 %	± 0,15 %	0,53	1,41 %	± 0,11 %	0,50	1,35 %	± 0,13 %	0,46
05	C	199	1291,29	4,06 %	± 0,13 %	0,90	3,56 %	± 0,18 %	0,87	3,55 %	± 0,18 %	0,86
06	C, D	50	555,43	0,12 %	± 0,04 %	0,09	0,10 %	± 0,04 %	0,09	0,10 %	± 0,04 %	0,09
07	C, D	75	909,68	0,65 %	± 0,12 %	0,18	0,54 %	± 0,11 %	0,18	0,48 %	± 0,13 %	0,17
08	C, D	100	865,94	0,31 %	± 0,09 %	0,43	0,21 %	± 0,08 %	0,41	0,15 %	± 0,06 %	0,40
09	C, D	150	1162,55	1,97 %	± 0,22 %	1,13	1,82 %	± 0,23 %	1,07	1,76 %	± 0,29 %	1,03
10	C, D	199	1395,85	2,94 %	± 0,19 %	1,83	2,60 %	± 0,22 %	1,75	2,18 %	± 0,16 %	1,66
11	C	120	1042,11	12,22 %	± 0,65 %	0,30	11,38 %	± 0,98 %	0,31	11,28 %	± 0,80 %	0,31
12	C	100	819,56	1,72 %	± 0,34 %	0,24	1,61 %	± 0,33 %	0,24	1,53 %	± 0,37 %	0,24
13	C, D	120	1541,14	1,95 %	± 0,13 %	0,78	1,71 %	± 0,10 %	0,75	1,59 %	± 0,09 %	0,72
14	C, D	100	866,37	0,32 %	± 0,13 %	0,36	0,17 %	± 0,10 %	0,35	0,11 %	± 0,07 %	0,33
Moyenne C				3,03 %	± 0,10 %	0,34	2,77 %	± 0,13 %	0,33	2,72 %	± 0,12 %	0,32
Moyenne C, D				1,18 %	± 0,05 %	0,68	1,02 %	± 0,05 %	0,66	0,91 %	± 0,05 %	0,63
Moyenne globale				2,11 %	± 0,05 %	0,51	1,90 %	± 0,07 %	0,49	1,81 %	± 0,06 %	0,47

6.6.3 Coopération entre les agents

Dans les deux sections précédentes, nous avons réalisé des expérimentations portant sur le processus de décision et les mécanismes d'apprentissage. Dans cette section nous nous focalisons sur la coopération entre les agents. Elle intervient via l'échange des meilleures solutions et lors du mimétisme. Pour mettre en évidence l'influence de cette coopération, et plus généralement de la distribution dans *CBM*, nous avons évalué les performances pour différentes tailles de la coalition. La figure 6.5 représente les traces d'exécution (en valeur moyenne) sur l'instance numéro 2 du benchmark Christofides et al. pour différentes tailles de coalition. Les différentes courbes correspondent à l'évolution de la fitness au cours de l'exécution. Quatre tailles de coalition sont considérées : 1, 10, 20 et 50 agents.

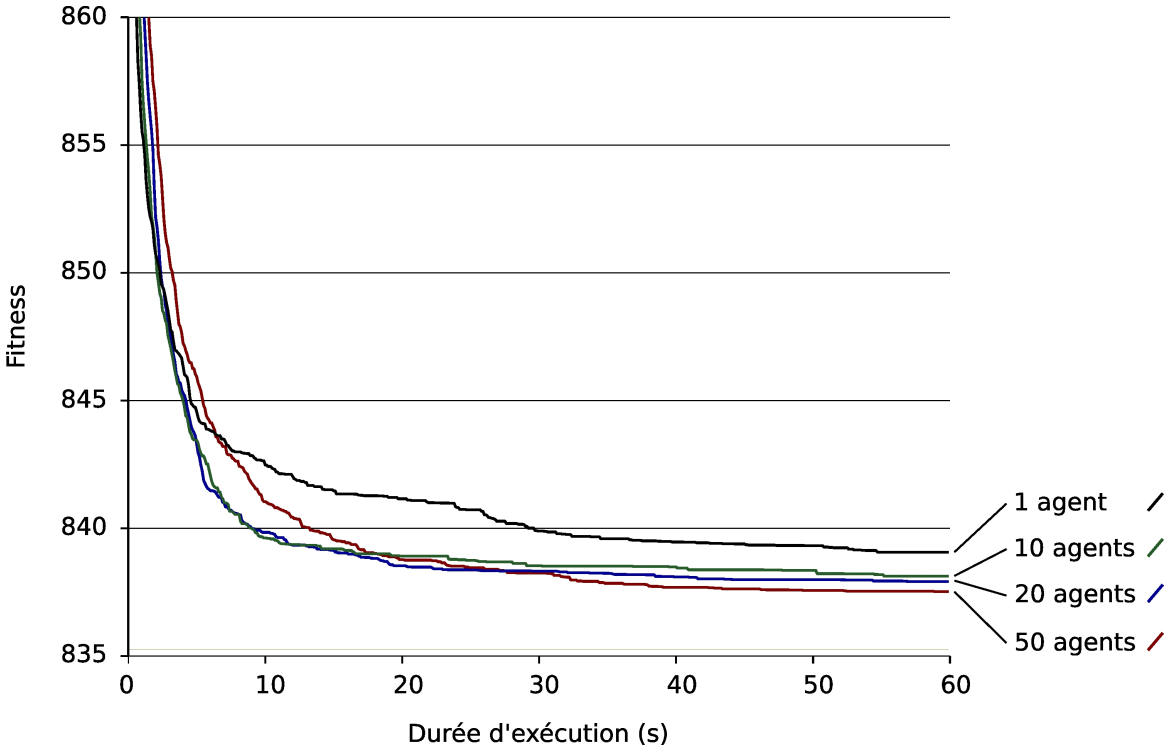

Figure 6.5: Trace d'exécution de *CBM* pour différentes tailles de coalition sur l'instance Christofides et al. n°2

On observe sur la figure 6.5 qu'à partir d'une certaine durée d'exécution, l'utilisation d'un nombre important d'agents permet d'améliorer nettement le résultat final. Lorsque l'on augmente le nombre d'agents, le processus d'optimisation est plus lent, cependant, les résultats sont meilleurs après un certain temps d'exécution. Par exemple, on observe que la configuration de *CBM* avec 50 agents fournit un meilleur résultat que les autres configurations à partir d'une durée d'exécution de 30 secondes. Le ralentissement du processus d'optimisation lorsque le nombre d'agents augmente peut s'expliquer par l'augmentation du temps nécessaire à l'exécution de l'ensemble des agents. Il faut noter que les temps d'exécution reportés dans la figure 6.5 correspondent aux temps cumulés d'exécution des agents. L'amélioration du résultat final lorsque le nombre d'agents augmente peut s'expli-

quer par une meilleure exploration de l'espace de recherche et par l'action plus efficace des mécanismes de coopération entre les agents. Ce résultat valide l'un des objectifs visé par *CBM* que nous avions précédemment formulé comme étant "l'obtention d'un système dans lequel l'exécution d'une coalition de C agents pour une durée T est plus performante que l'exécution d'un unique agent pendant une durée $C * T$".

6.7 Comparaison de *CBM* avec d'autres métaheuristiques

Dans cette section, *CBM* est comparée à quatre autres approches heuristiques parmi les plus performantes sur le *VRP* : *GTS* [Toth and Vigo, 2003], *UTSA* [Cordeau et al., 2004], *AGES* [Mester and Bräysy, 2005, 2007] et un algorithme mémétique [Prins, 2004].

Pour effectuer une comparaison des temps d'exécution, ces derniers ont été normalisés en utilisant les facteurs proposés par Dongarra [Dongarra, 2006]. Ces facteurs sont déterminés par les performances des ordinateurs mesurées en Mflop/s (millions d'opérations à virgule flottante par seconde). Ils ont été obtenus à partir de l'exécution du benchmark *Linpack*[1]. Le tableau 6.4 reporte les valeurs de performance des ordinateurs utilisés pour l'exécution des différentes approches. À partir de ces valeurs, il est possible de rapporter les temps d'exécution de chaque approche à un ordinateur de référence. Ici, il s'agit de notre Pentium IV 3GHz utilisé pour l'exécution de *CBM*.

Approche	Ordinateur	Performance estimée suivant [Dongarra, 2006] en Mflop/s	Facteur de normalisation du temps d'exécution
CBM	Pentium IV 3GHz	1571	1
GTS	Pentium 200MHz	38	38/1571
UTSA	Pentium IV 2GHz	781	781/1571
AGES	Pentium IV 2GHz	782	1317/782
Alg. mémétique	Pentium III 1GHz	234	234/1571

Table 6.4: Facteurs de performance sur le benchmark *Linpack*, tiré de [Dongarra, 2006]

Pour les benchmarks [Christofides et al., 1979] et [Golden et al., 1998], *CBM* a été paramétrée avec les valeurs présentées dans le tableau 6.5. Ce paramétrage a été déterminé après une série de tests effectués sur quelques instances tirées des deux benchmarks. Il a été conservé pour l'ensemble des expérimentations effectuées.

Les tableaux 6.6 et 6.7 présentent les résultats des différentes approches sur les benchmarks de Christofides et al. [1979] et Golden et al. [1998] respectivement. Les quatre premières colonnes rappellent les caractéristiques des instances ainsi que les meilleures solutions connues. Dans les deux premières colonnes des tableaux correspondant à *CBM* sont

[1]Le benchmark *Linpack*, introduit par Dongarra permet de mesurer la puissance en temps de calcul d'un ordinateur à partir de l'exécution d'un programme de résolution d'équations linéaires. (`http://www.netlib.org/linpack/`)

Paramètre	Description	Valeur
C	Taille de la coalition	20
α	Poids initial	1,0
σ_1	Valeur de renforcement faible	0,5
σ_2	Valeur de renforcement forte	1,0
ρ	Coefficient d'imitation	0,3

Table 6.5: Paramétrages de *CBM* pour les benchmarks Christofides et al. et Golden et al.

indiqués la valeur moyenne obtenue sur 10 exécutions, puis l'écart relatif à la meilleure solution connue. La troisième colonne relative à *CBM* indique l'intervalle de confiance à 95% des écarts, calculé sur les dix exécutions. La dernière colonne donne la durée moyenne d'exécution en minutes obtenue sur un Pentium IV 3GHz. Pour les quatre autres approches sont donnés l'écart à la meilleure solution ainsi que la durée d'exécution. Excepté dans la dernière ligne, les durées sont spécifiques aux ordinateurs utilisés : Pentium 200MHz pour *GTS*, Pentium IV 2GHz pour *UTSA* et pour *AGES*, et un Pentium III 1GHz pour l'algorithme mémétique. Les durées moyennes normalisées suivant le tableau 6.4 et rapportées à un Pentium IV 3GHz sont indiquées uniquement dans la dernière ligne des tableaux.

Nous avons cherché à obtenir une comparaison aussi fiable et précise que possible. Néanmoins, quelques remarques s'imposent. Tout d'abord, les temps d'exécution sont des indicateurs assez approximatifs. En effet, les méthodes ont été implémentées suivant différents langages de programmation et exécutées sur différents matériels. Les facteurs de performance tirés de [Dongarra, 2006] sont issus d'informations assez partielles sur la configuration des ordinateurs utilisés. Ensuite, tous les résultats n'ont pas été obtenus nécessairement suivant le même protocole d'expérimentation. Si dans l'approche par algorithme mémétique [Prins, 2004], l'auteur indique clairement qu'un seul paramétrage par benchmark a été utilisé, certains auteurs n'indiquent pas si les résultats sont issus ou non de plusieurs paramétrages. Enfin, il est parfois difficile de savoir si les valeurs présentées sont des valeurs moyennes sur plusieurs exécutions. En dépit de ces différentes remarques, nous pensons que les ordres de grandeur des performances sont néanmoins représentés. Les facteurs de Dongarra conduisent à une approximation assez large, mais néanmoins vérifiable. Nous pouvons ainsi mieux situer l'approche *CBM* dans le panorama des métaheuristiques.

Sur les deux benchmarks, l'approche *AGES* donne les meilleurs résultats en termes de qualité de solution obtenue. Pour cette approche, le meilleur résultat connu est atteint sur toutes les instances sauf sur l'instance numéro 10 du benchmark de Christofides et al. [1979]. Les temps d'exécution pour *AGES* semblent assez longs et l'algorithme mémétique semble fournir le meilleur compromis entre les temps d'exécution et la qualité des solutions.

Les résultats de *CBM* sont comparables aux autres approches, avec des écarts moyens aux meilleures solutions de 0,52% sur le benchmark de Christofides et al. [1979] et de 1,64% sur le benchmark de Golden et al. [1998]. Ces résultats sont prometteurs en considé-

rant que les implémentations d'opérateurs peuvent être améliorées. Par exemple en restreignant la taille des voisinages comme dans les approches *GTS* et *UTSA* ou en utilisant un codage des solutions plus performant comme dans l'algorithme mémétique de Prins [2004], il semble possible de réduire substantiellement les temps d'exécution.

En plus de la comparaison des approches en termes de qualité et de temps d'exécution, il est possible de considérer d'autres critères parfois plus subjectifs d'évaluation. Dans [Cordeau et al., 2002], les auteurs proposent deux critères supplémentaires de comparaison : la simplicité et la flexibilité. La simplicité est relative à la facilité de compréhension et d'implantation d'un algorithme tandis que la flexibilité est relative à la facilité d'adaptation d'un algorithme pour traiter de nouvelles contraintes ou de nouveaux problèmes. Ainsi, l'approche *AGES* qui est la plus performante sur les benchmarks considérés, est également considérée comme étant la plus complexe des heuristiques. Pour ce qui est de *CBM*, la flexibilité est l'un des objectifs visé. En adoptant une architecture générique suivant le principe des hyper-heuristiques et de la coalition, nous souhaitons faciliter l'étape de spécialisation.

En résumé, *CBM* se distingue des autres approches suivant trois critères essentiels. Premièrement, l'approche est fondée sur le paradigme multi-agent et en cela les agents ont de l'intentionnalité, c'est-à-dire des capacités de perception, de délibération et d'action. Deuxièmement, ces capacités sont mises en œuvre notamment via les mécanismes de décision et d'apprentissage permettant l'amélioration du choix des opérateurs. Enfin, troisièmement, l'organisation en coalition donne la possibilité de distribuer l'exécution à un niveau élevé de granularité, correspondant à des stations de travail en réseau.

Table 6.6: Résultats sur le benchmark Christofides et al.

Instance				CBM				GTS		UTSA		AGES		Alg. mémétique	
	Contrainte	Taille	Meilleure connue	Moyenne	Écart	IC (95%)	Durée (min)[a]	Écart	Durée (min)[b]	Écart	Durée (min)[c]	Écart	Durée (min)[d]	Écart	Durée (min)[e]
1	C	50	524,61	524,61	0,00%	± 0,00%	0,00	0,00%	0,81	0,00%	2,32	0,00%	0,01	0,00%	0,00
2	C	75	835,26	837,00	0,21%	± 0,23%	2,38	0,40%	2,22	0,00%	14,78	0,00%	0,26	0,00%	0,77
3	C	100	826,14	827,55	0,17%	± 0,10%	2,47	0,29%	2,39	0,00%	11,67	0,00%	0,05	0,00%	0,46
4	C	150	1028,42	1034,47	0,59%	± 0,14%	3,00	0,47%	4,51	0,41%	26,66	0,00%	0,47	0,31%	5,50
5	C	199	1291,29	1316,08	1,92%	± 0,36%	9,00	2,09%	7,50	1,90%	57,68	0,00%	101,93	0,69%	19,10
6	C, D	50	555,43	555,43	0,00%	± 0,00%	0,07	0,00%	0,86	0,00%	3,03	0,00%	0,02	0,00%	0,01
7	C, D	75	909,68	910,32	0,07%	± 0,11%	0,99	1,21%	2,75	0,00%	7,41	0,00%	0,43	0,29%	1,42
8	C, D	100	865,94	865,94	0,00%	± 0,00%	0,61	0,41%	2,90	0,00%	10,93	0,00%	0,44	0,00%	0,37
9	C, D	150	1162,55	1176,59	1,21%	± 0,37%	3,00	0,91%	5,67	0,46%	51,66	0,00%	1,22	0,15%	7,25
10	C, D	199	1395,85	1419,21	1,67%	± 0,45%	5,00	2,86%	9,11	1,50%	106,28	0,38%	2,45	1,74%	26,83
11	C (c)	120	1042,11	1045,65	0,34%	± 0,64%	4,65	0,07%	3,18	3,01%	11,67	0,00%	0,05	0,00%	0,30
12	C (c)	100	819,56	819,56	0,00%	± 0,00%	0,54	0,00%	1,10	0,00%	9,02	0,00%	0,01	0,00%	0,05
13	C, D (c)	120	1541,14	1558,11	1,10%	± 0,38%	5,00	0,28%	9,34	0,53%	21,00	0,00%	0,63	0,12%	10,44
14	C, D (c)	100	866,37	866,37	0,00%	± 0,00%	0,37	0,00%	1,41	0,00%	10,53	0,00%	0,08	0,00%	0,09
Moyenne globale					0,52%	± 0,07%	2,65	0,64%	3,84	0,56%	24,62	0,03%	7,72	0,24%	5,19
Durée normalisée							2,65		0,09		12,24		4,58		0,77

[a] Durées sur un Pentium IV 3GHz
[b] Durées sur un Pentium 200MHz
[c] Durées sur un Pentium IV 2GHz
[d] Durées sur un Pentium IV 2GHz
[e] Durées sur un Pentium III 1GHz

Table 6.7: Résultats sur le benchmark Golden et al.

Instance				CBM				GTS		UTSA		AGES		Alg. mémétique	
	Contrainte	Taille	Meilleure connue	Moyenne	Écart	IC (95%)	Durée (min)[a]	Écart	Durée (min)[b]	Écart	Durée (min)[c]	Écart	Durée (min)[d]	Écart	Durée (min)[e]
1	C	240	5627,54	5805,28	3,16%	± 0,96%	40	1,93%	5	0,97%	10	0,00%	9	0,34%	32
2	C	320	8447,92	8649,13	2,38%	± 0,05%	54	1,24%	8	2,48%	35	0,00%	47	0,00%	78
3	C	400	11036,22	11291,18	2,31%	± 0,09%	40	3,32%	13	0,01%	55	0,00%	41	0,00%	121
4	C	480	13624,52	13650,15	0,19%	± 0,08%	40	9,44%	15	0,85%	83	0,00%	470	0,00%	188
5	C	200	6460,98	6460,98	0,00%	± 0,00%	2	3,66%	2	4,57%	5	0,00%	0	0,00%	1
6	C	280	8412,80	8413,09	0,00%	± 0,00%	40	6,54%	5	1,48%	19	0,00%	75	0,00%	10
7	C	360	10195,56	10205,06	0,09%	± 0,06%	40	3,45%	12	0,70%	26	0,00%	3	0,00%	39
8	C	440	11663,55	11865,31	1,73%	± 0,15%	40	3,20%	11	1,77%	71	0,00%	34	1,42%	88
9	C, D	255	583,39	593,1	1,67%	± 0,17%	40	1,71%	12	0,69%	37	0,00%	8	1,40%	14
10	C, D	323	742,03	756,61	1,96%	± 0,75%	60	1,30%	16	1,45%	51	0,00%	6	1,26%	37
11	C, D	399	918,45	941,87	2,55%	± 0,57%	57	1,92%	33	1,16%	42	0,00%	110	1,59%	79
12	C, D	483	1107,19	1151,19	3,97%	± 0,37%	60	3,61%	43	1,11%	157	0,00%	600	2,40%	31
13	C, D	252	859,11	867,84	1,02%	± 0,32%	48	1,13%	11	1,95%	35	0,00%	10	1,87%	15
14	C, D	320	1081,31	1095,68	1,33%	± 0,37%	50	1,38%	15	1,92%	22	0,00%	1	0,46%	34
15	C, D	396	1345,23	1371,51	1,95%	± 0,29%	52	1,80%	18	1,38%	58	0,00%	7	1,65%	110
16	C, D	480	1622,69	1655,71	2,03%	± 0,29%	56	1,83%	23	1,50%	130	0,00%	20	1,74%	131
17	C, D	240	707,79	712,21	0,63%	± 0,10%	40	0,46%	14	0,44%	18	0,00%	1	0,37%	6
18	C, D	300	998,73	1015,38	1,67%	± 0,62%	51	1,81%	21	1,59%	67	0,00%	3	1,61%	39
19	C, D	360	1366,86	1393,12	1,92%	± 0,19%	40	2,49%	30	1,24%	66	0,00%	6	0,70%	74
20	C, D	420	1821,15	1862,02	2,24%	± 0,48%	56	5,20%	43	1,82%	135	0,00%	8	1,39%	210
Moyenne globale					1,64%	± 0,08%	45	2,87%	18	1,45%	56	0,00%	73	0,91%	67
Durée normalisée							45		0,42		28		43		10

[a] Durées sur un Pentium IV 3GHz
[b] Durées sur un Pentium 200MHz
[c] Durées sur un Pentium IV 2GHz
[d] Durées sur un Pentium IV 2GHz
[e] Durées sur un Pentium III 1GHz

6.8 Bilan

Dans ce chapitre a été présentée l'application de *CBM* à un problème de tournée de véhicules. L'approche *CBM* semble pouvoir être adaptée aisément à la résolution de ce type de problème. La complexité du problème justifiant une approche de voisinage, l'accent est mis dans *CBM* sur une utilisation adéquate des nombreux opérateurs de voisinages pouvant être définis. Un ensemble d'expérimentations ont été menées afin d'analyser point par point l'apport des différents composants de *CBM*. Ensuite, une comparaison des résultats obtenus avec ceux d'autres approches a été réalisée.

À travers cette application, nous avons tout d'abord souhaité illustrer la démarche de spécialisation dans *CBM*. L'adoption dans *CBM* d'une approche inspirée des hyper-heuristiques permet de ne spécialiser que les opérateurs d'intensification et de diversification. Le choix de ces opérateurs est adapté dynamiquement par apprentissage. Les premières expérimentations ont permis de valider l'apport de ces mécanismes d'adaptation, mais aussi de confirmer l'intérêt de la coopération entre les agents.

Les résultats obtenus avec *CBM* sur deux benchmarks représentatifs du domaine indiquent que *CBM* reste compétitive vis-à-vis des meilleures approches. En dépit d'une implantation assez directe des opérateurs, nous restons en moyenne sous la barre des 2%, ce qui équivaut à une méthode classique telle que *UTSA*. L'objectif recherché avec *CBM* n'est pas seulement de proposer une approche efficace sur un problème particulier, mais de fournir aussi une méthode flexible qui exploite les avantages des mécanismes d'adaptation des choix d'opérateurs et de la distribution du calcul. Pour renforcer la présentation de l'aspect générique de *CBM*, nous proposons dans le chapitre suivant une deuxième application portant sur un problème de positionnement.

CHAPITRE 7

APPLICATION DE *CBM* AU PROBLÈME DE POSITIONNEMENT

Sommaire

7.1 Introduction

Ce chapitre est consacré à l'étude de l'application de *CBM* à la résolution d'un deuxième problème d'optimisation. Le problème traité est un problème de positionnement discret sans contrainte de capacité, appelé *UFLP* (*Uncapacitated Facility Location Problem*). Le but recherché en expérimentant à nouveau la démarche de spécialisation est d'insister sur la possibilité offerte d'un développement générique, mettant en jeu des composants simples et produisant au final une heuristique distribuée, robuste, et compétitive. C'est la conjonction de ces divers facteurs considérés suivant le paradigme multi-agent et pris ensemble qui constituent le cœur de la méthode et qui doivent justifier et motiver le choix de son utilisation.

En nous focalisant tout d'abord sur la démarche de spécialisation, nous précisons comment est introduite la connaissance du problème via la conception d'opérateurs simples de voisinage. Ensuite, des évaluations comparatives avec des approches connues les plus performantes sur le problème sont présentées. En nous appuyant sur l'utilisation de jeux de tests de grande taille, considérés comme représentatifs de la difficulté du problème, nous établirons un panorama comparatif des approches prenant en compte la qualité des solutions et le temps d'exécution. Nous chercherons à montrer, qu'en dépit de sa simplicité et de sa généricité, l'approche *CBM* n'en est pas moins compétitive vis-à-vis des approches les plus performantes.

La première section de ce chapitre dresse un état de l'art des différents problèmes de positionnement en rappelant leur intérêt pour le domaine des transports terrestres. Une définition précise de *UFLP* est donnée. Ensuite, la deuxième section décrit les principales méthodes de résolution existantes. Dans la section suivante 7.4, la démarche de spécialisation de *CBM* est détaillée. Les opérateurs de bas niveau d'intensification et de diversification sont décrits. Ensuite, deux sections présentent les tests empiriques réalisés. La section 7.5 introduit les deux benchmarks utilisés dans les expérimentations. Enfin, la section 7.6 est consacrée à l'exposé des résultats obtenus avec *CBM*. L'approche est comparée à des heuristiques et métaheuristiques récentes appliquées sur les deux benchmarks. Un bilan est donné dans la dernière section.

7.2 Le problème de positionnement

Les problèmes de positionnement se définissent de manière générale par l'intitulé suivant :

Définition 7.1 Définition générale d'un problème de positionnement

Définir la position d'éléments (*facilities*) permettant de servir des clients ou une demande afin d'optimiser un ou plusieurs objectifs.

Cette définition regroupe une large gamme de problèmes de positionnement pouvant se distinguer par le type de facility[1], l'espace de positionnement, le type de client ou de demande et la fonction à optimiser [Moujahed et al., 2007]. Nous précisons ces caractéristiques.

Type de facility : En pratique les facilities à positionner peuvent être de diverses natures : des entrepôts, des arrêts de bus ou encore des antennes pour les télécommunications. Dans la formulation des problèmes, cela se traduit par des différences sur le nombre de facilities à positionner (une ou plusieurs) et par des différences sur le type de distribution et la nature des services fournis (homogène ou hétérogène). Dans certaines versions dites multi-niveaux, les facilities d'un niveau particulier deviennent des points de demande relativement à un niveau supérieur. On obtient ainsi une hiérarchie de points de regroupement pouvant être desservis par des falicities de taille et de capacité variables.

Espace de positionnement : On distingue globalement trois types de possibilités pour le positionnement de facilities : cas discret, cas continu ou en réseau. Dans un espace de positionnement discret, un ensemble de positions potentielles sont définies préalablement sur le plan et le positionnement consiste à ouvrir ou fermer ces sites. Dans le cas continu, les positions sont déterminées par des variables continues représentant des coordonnées dans le plan [Moujahed et al., 2007]. Dans les modèles de positionnement sur un réseau, le problème est défini dans les graphes. Les positions potentielles sont des nœuds du graphe et les coûts, non nécessairement euclidiens, sont associés aux arêtes du graphe.

Type de client : La demande ou les clients sont également représentés de manière discrète ou continue. De la même manière que les facilities, les clients peuvent constituer un ensemble homogène ou hétérogène avec différentes caractéristiques.

Fonction à optimiser : La fonction à minimiser est dans la plupart des cas le coût d'ouverture des facilities et/ou la distance définie entre les facilities et les clients. Cependant, un grand nombre de variantes sont possibles prenant en compte le degré de couverture ou encore la dispersion des facilities [Current et al., 2004].

Une classification plus complète des problèmes de positionnement a été introduite dans [Hamacher et al., 1996]. Cette classification considère cinq critères : (i) le nombre et le type de facility, (ii) le type d'espace de positionnement, (iii) le type de contraintes s'appliquant aux positionnement des facilities, (iv) le type de coût, (v) et le type de fonction objectif. À partir de ces critères, la plupart des problèmes de positionnement peuvent être définis.

L'engouement de la recherche opérationnelle pour les problèmes de positionnement s'explique par les enjeux économiques sous-jacents et la difficulté de résolution optimale

[1] Il ne semble pas y avoir de terme équivalent au mot anglais *facility*. Le mot français qui semble le plus proche est "service". Mais comme ce terme français peut désigner aussi bien l'élément en soit, et le lien entre l'élément et le client nous utiliserons le terme anglais.

de ces problèmes. En effet, ces problèmes sont $\mathcal{NP}$-complets et donc difficiles à résoudre en particulier s'il s'agit de grandes instances [Current et al., 2004]. Ces problèmes ont été largement étudiés en recherche opérationnelle depuis les années 60. Encore aujourd'hui, ces problèmes de positionnement font l'objet d'une attention toute particulière dans des domaines d'application tels que les télécommunications ou encore les réseaux de capteurs.

Dans ce chapitre, nous nous intéressons au problème de positionnement discret sans contrainte de capacité *UFLP* (*Uncapacitated Facility Location Problem*). Ce problème consiste à déterminer les positions des facilities relativement à un ensemble de sites potentiels. Ainsi, une solution correspond aux sites ouverts. L'objectif est de minimiser les distances entre les clients et les sites ouverts les plus proches, sachant que l'ouverture d'un site engendre un coût. La figure 7.1 présente un exemple d'instance du problème avec une solution associée. L'instance est composée de quatre sites et de neuf clients. Dans la solution représentée deux des quatre sites sont ouverts. Le coût de cette solution correspond à la somme des coûts d'affectation de chaque client au site ouvert le plus proche, additionnée de la somme des coûts d'ouverture des sites. Une formulation précise du problème est donnée ci-dessous.

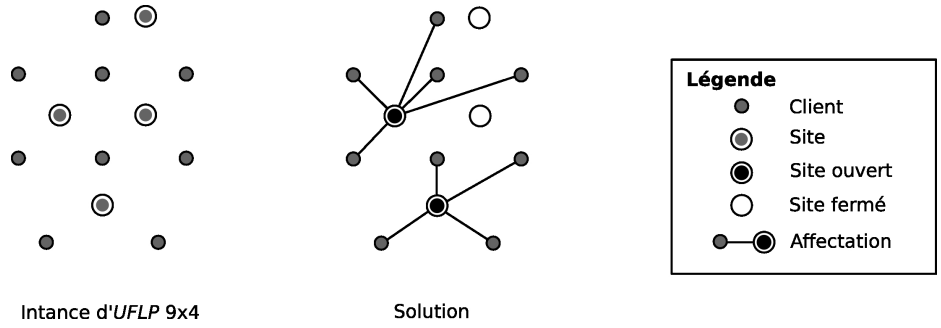

Figure 7.1: Schéma d'une instance et d'une solution d'*UFLP*

Soit $I = i_1, ..., i_m$ un ensemble de sites où peuvent être placées les facilities, et $J = j_1, ..., j_n$ un ensemble de clients à desservir. Pour chaque site $i \in I$, est défini un coût de mise en service f_i, et pour chaque couple site/client est défini un coût d'affectation c_{ij}. Le problème de positionnement sans contrainte de capacité (*UFLP*) consiste à trouver un sous-ensemble S de sites à ouvrir de manière à ce que le coût $C(S)$ soit minimisé. Ce coût correspond à la somme des coûts de mise en service additionnée à la somme des coûts d'affectation entre chaque client et l'installation ouverte la plus proche. Le problème *UFLP* s'écrit :

$$\text{Minimiser } C(S), \emptyset \subset S \subseteq I \tag{7.1}$$

avec,

$$C(S) = \sum_{i \in S} f_i + \sum_{j \in J} min\{c_{ij} | i \in S\} \tag{7.2}$$

Il a été montré que l'*UFLP* est un problème $\mathcal{NP}$-dur. La preuve est donnée dans [Cornuéjols et al., 1990].

7.3 Approches de résolution existantes

Plusieurs méthodes exactes ont été développées pour l'*UFLP*. Une revue des différentes méthodes exactes est présentée dans [Cornuéjols et al., 1990]. Cependant, des méthodes comme l'approche dual [Erlenkotter, 1978] ou primal-dual [Körkel, 1989] ne permettent que la résolution d'instances de petite taille vérifiant certaines propriétés. Par exemple, dans le benchmark des instances de grande taille présenté dans la suite du chapitre, une seule instance sur 90 a été résolue de manière optimale. Cette difficulté de résolution justifie l'emploi de méthodes heuristiques.

La première méthode heuristique appliquée à l'*UFLP* semble être celle de Kuehn et Hamburger [Kuehn and Hamburger, 1963]. Elle est constituée d'une phase de construction suivie d'une phase de recherche locale. La construction d'une solution débute avec tous les sites fermés, et consiste à ouvrir les sites produisant une diminution maximum du coût. Cette phase de construction se termine lorsqu'il n'existe plus de site pouvant être ouvert conjointement avec une réduction du coût de la solution. La seconde phase est une recherche locale dans laquelle le voisinage d'une solution consiste à fermer et à ouvrir simultanément deux sites.

Parmi les méthodes les plus performantes ayant ensuite été proposées, Gosh a développé une méthode de recherche tabou et aussi une méthode nommée *CLM* pour "*Complete Local search with Memory*" [Ghosh, 2003]. Cette seconde méthode utilise les mêmes structures de voisinage que la première, mais avec une procédure de recherche différente.

La méthode de recherche tabou de Ghosh est fondée sur le voisinage *2-swap* qui considère, lors d'un mouvement d'une solution vers une solution voisine, soit l'ouverture ou la fermeture d'un site, soit l'ouverture et la fermeture simultanée de deux sites. La liste tabou est constituée des sites ayant récemment été mis en jeux lors d'un mouvement dans le voisinage. Le statut (ouvert/fermé) des sites présents dans cette liste tabou ne peut pas être modifié lors des prochains mouvements dans le voisinage. En plus de cette mémoire à court terme, la procédure de recherche tabou utilise une mémoire à moyen terme appelée aussi mémoire de fréquence (*frequency-based memory*) qui enregistre pour chaque site le nombre de changements de statuts intervenus lors de la recherche. A partir de cette mémoire, une pénalité est appliquée sur la fonction de coût afin d'éviter une recherche restreinte sur une petite partie de l'espace de solutions.

CLM est une approche inspirée des méthodes de recherche dans un graphe. Chaque solution est considérée comme un nœud, et les nœuds voisins sont obtenus par le voisinage *2-swap*. Dans cette méthode, le principe consiste à conserver la trace de chaque solution visitée dans une liste intitulée *DEAD* et de placer les solutions permettant de continuer la

recherche dans une liste nommée *LIVE*. Cette méthode utilise en plus une procédure de recherche locale afin d'accélérer la recherche.

Suite à ces travaux, Resende et Werneck ont proposé une méthode hybride [Resende and Werneck, 2003] combinant des concepts provenant de la recherche dispersée, de la recherche tabou et des algorithmes évolutionnistes. Le principe consiste à faire évoluer une population de solutions prometteuses. À chaque génération, un ensemble de solutions est généré aléatoirement. Ces solutions sont ensuite améliorées par une recherche locale utilisant le voisinage *2-swap* puis en appliquant une procédure de *path relinking* avec les solutions prometteuses. Cette procédure utilisée suivant le cadre de la recherche dispersée consiste à construire le chemin entre deux solutions suivant un voisinage donné et à retenir la meilleure solution générée. Finalement, l'ensemble des solutions prometteuses est mis à jour avec les solutions obtenues par la recherche locale et l'utilisation du *path relinking*. La mise à jour est basée sur les critères de distance et de coût pour sélectionner les solutions. Ainsi, la sélection favorise non seulement les meilleures solutions (coût minimum) et favorise également une certaine diversité de solutions.

Plus récemment, Sun à proposé une recherche tabou qui semble être la métaheuristique la plus performante à l'heure actuelle sur l'*UFLP* [Sun, 2006]. Cette méthode reprend les principes de la recherche tabou de Ghosh mais en utilisant un voisinage de type *1-switch*. Ce voisinage modifie le statut d'un seul site lors d'un mouvement. Ensuite, Sun introduit une mémoire à long terme en plus des mémoires à court et moyen terme déjà utilisées par Ghosh. La procédure associée à cette nouvelle mémoire vise à favoriser l'exploration de nouvelles zones de l'espace de recherche.

D'autres métaheuristiques moins performantes mais présentant des traits originaux ont été développées pour l'*UFLP*. Sevkli et Guner ont développé une approche par essaims particulaires [Sevkli and Guner, 2006], [Guner and Sevkli, 2008]. Dans cette approche, la position de chaque particule est définie dans un espace à n dimensions. Chacune des dimensions correspond à un site. Les auteurs définissent une fonction permettant d'obtenir les données discrètes d'une solution à partir de la position d'une particule. En plus d'appliquer une procédure standard pour faire évoluer les particules, les auteurs utilisent également une procédure de recherche locale pour tenter d'améliorer la meilleure solution connue de l'essaim.

Dans [Emin Aydin and Fogary, 2004] les auteurs présentent une approche de recuit simulé distribué pour traiter l'*UFLP*. La démarche possède des similitudes avec notre approche *CBM*, dans la mesure où une procédure de recuit simulé est appliquée de manière distribuée sur différentes sous-populations à la manière de la méthode des îles [Tanese, 1989]. Le voisinage utilisé est de type *2-swap*. La méthode est appliquée à deux problèmes d'optimisation : un problème d'ordonnancement (*Job-shop scheduling problem*) et l'*UFLP*.

7.4 Spécialisation de *CBM* pour la résolution du problème de positionnement

Tout comme pour le *VRP* la spécialisation de *CBM* pour traiter l'*UFLP* consiste à définir un ensemble d'opérateurs d'intensification et de diversification. Deux opérateurs d'intensification et 5 opérateurs de diversification ont été définis. Ils sont détaillés dans les deux sections suivantes.

7.4.1 Opérateurs d'intensification pour l'*UFLP*

Les deux opérateurs d'intensification sont des procédures de recherche locale. Ils suivent le même schéma général de recherche locale proposé dans l'algorithme 5 du chapitre précédent.

Le voisinage utilisé par le premier opérateur est de type *1-switch*. Cette structure de voisinage consiste à ouvrir ou fermer un site pour obtenir une solution voisine. Il s'agit du voisinage utilisé dans la recherche tabou de Sun [Sun, 2006]. La figure 7.2 représente un mouvement dans le voisinage *1-switch* d'une solution. La seconde solution est obtenue en ouvrant le site numéro 3.

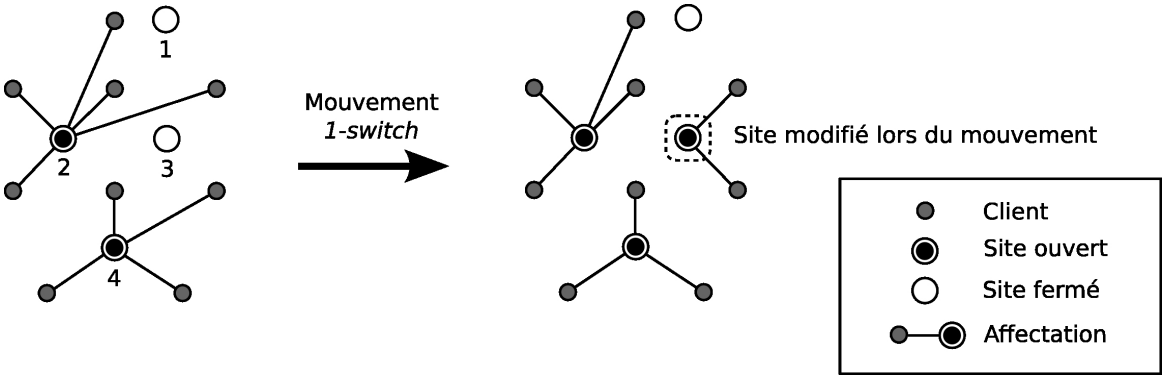

Figure 7.2: Mouvement dans le voisinage *1-switch* d'une solution

Le second opérateur de recherche locale est une version modifiée du voisinage *2-swap* utilisé dans les approches de Ghosh [Ghosh, 2003] ainsi que de Resende et Werneck [Resende and Werneck, 2003]. Dans ce voisinage, un mouvement est réalisé par la fermeture et l'ouverture simultanée de deux sites. La figure 7.3 représente un mouvement réalisé dans le voisinage *2-swap* d'une solution. La seconde solution est obtenue en fermant le site numéro 2 et en ouvrant le site numéro 1 de la solution initiale.

7.4.2 Opérateurs de diversification pour l'*UFLP*

Cinq opérateurs de diversification sont proposés pour traiter l'*UFLP* : deux opérateurs de génération, deux opérateurs de mutation et un opérateur de croisement.

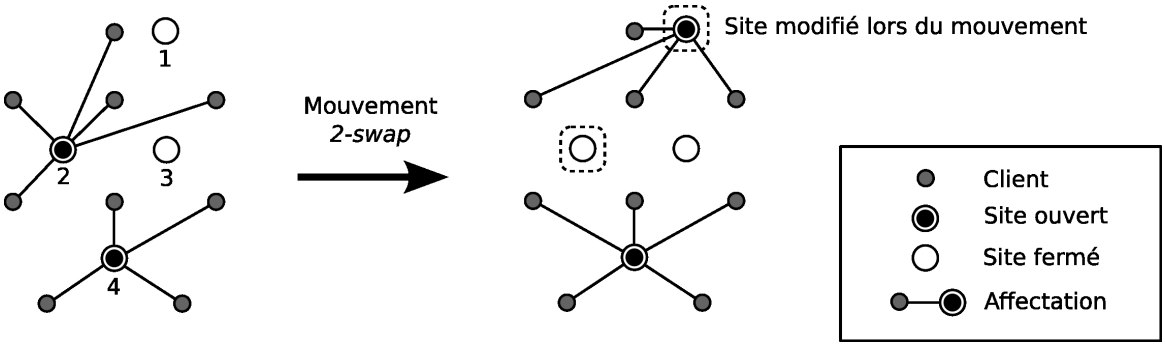

Figure 7.3: Mouvement dans le voisinage *2-swap* d'une solution

Le premier opérateur de génération suit deux étapes pour créer une solution. Dans un premier temps, un nombre aléatoire de sites sont ouverts, puis les sites auxquels aucun client n'est associé sont fermés.

Le second opérateur de génération est une procédure "gloutonne". La construction d'une solution débute avec l'ensemble des sites fermés. Ensuite, les sites sont progressivement ouverts dans un ordre aléatoire avec la condition qu'ils améliorent le coût de la solution. La procédure s'arrête lorsqu'il n'existe plus de site à ouvrir permettant d'améliorer le coût de la solution.

Les deux opérateurs de mutation correspondent à une même procédure paramétrée à deux taux de mutation différents. Cette procédure consiste à effectuer une série de mouvement aléatoire dans le voisinage *1-switch*. Les taux de mutation correspondent au rapport entre le nombre de mouvements effectués et le nombre de sites dans l'instance du problème.

L'opérateur de croisement est un croisement uniforme. Pour obtenir une solution, cette procédure choisit de manière équiprobable le statut des sites entre les deux solutions initiales. La figure 7.4 présente un exemple de croisement uniforme pour une instance d'*UFLP* avec quatre sites. Dans cet exemple, la solution obtenue après croisement reprend le statut des sites 2 et 3 de la première solution, combinés aux statuts des sites 1 et 4 de la seconde solution.

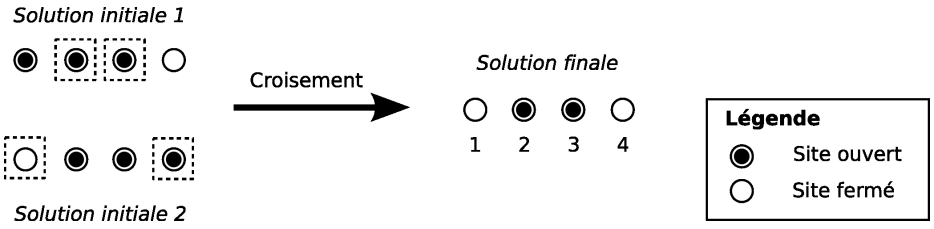

Figure 7.4: Croisement uniforme de deux solutions d'*UFLP*

7.5 Présentation des benchmarks

Nous avons sélectionné deux benchmarks pour évaluer *CBM* sur l'*UFLP*. Ces benchmarks ont été choisis afin de disposer d'un panel varié d'instances, mais aussi parce qu'ils constituent des problèmes de référence dans la littérature.

Le premier benchmark est issu de la suite de benchmarks *OR-Library*[2]. Il est constitué de 15 instances de petite et moyenne tailles allant de 50×16 (nombre de clients $\times$ nombre de sites) à 1000×100. L'ensemble des instances du benchmark a été résolu de manière exacte [Ghosh, 2003]. Ces instances ont été proposées à l'origine pour un problème de positionnement avec contrainte de capacité dans [Beasley, 1993] mais ont été ensuite adaptées à l'*UFLP*. Les caractéristiques des instances du benchmark sont rapportées dans le tableau 7.1.

Instance	Taille	Valeur de l'optimum
cap71	50×16	932 615,75
cap72	50×16	977 799,40
cap73	50×16	1 010 641,45
cap74	50×16	1 034 976,98
cap101	50×25	796 648,44
cap102	50×25	854 704,20
cap103	50×25	893 782,11
cap104	50×25	928 941,75
cap131	50×50	793 439,56
cap132	50×50	851 495,33
cap133	50×50	893 076,71
cap134	50×50	928 941,75
capa	1000×100	17 156 454,48
capb	1000×100	12 979 071,58
capc	1000×100	11 505 594,33

Table 7.1: Caractéristiques des instances du benchmark OR-Library pour l'*UFLP*

Le second benchmark nommé *Körkel-Ghosh* est tiré de l'*UflLib*[3]. Il a été proposé dans [Ghosh, 2003] en suivant le principe de génération d'instances énoncé dans [Körkel, 1989]. Le benchmark est composé de 90 instances allant d'une taille de 250×250 à une taille de 750×750. Les instances de ce benchmark n'ont pas été résolues de manière exacte. Elles sont divisées en deux groupes de 45 instances avec, d'une part, des instances symétriques ($c_{ij} = c_{ji}$), et d'autre part, des instances asymétriques ($c_{ij} \neq c_{ji}$). Les instances sont générées avec des coûts de transport c_{ij} tirés aléatoirement dans un intervalle $[1000, 2000]$ et

[2] *OR-Library* est un ensemble de données tests relatif aux problèmes d'optimisation combinatoire les plus répendus. Ces données sont maintenues par Beasley et accessible sur son site : `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[3] *UflLib* est une collection de benchmarks pour l'*UFLP* regroupés sur le site : `http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/`

les coûts de mise en service f_i sont répartis en trois classes : la classe A avec des coûts dans l'intervalle $[100, 200]$, la classe B $[1000, 2000]$, et la classe C $[10000, 20000]$. Ainsi, les différentes classes correspondent à des instances avec respectivement des coûts de mise en service faibles (classe A), moyens (classe B) et forts (classe C). Pour chaque combinaison de caractéristiques, cinq instances ont été générées. Ces différentes combinaisons sont résumées dans le tableau 7.2. La dernière colonne du tableau présente les meilleurs résultats connus, obtenus par recherche tabou dans [Sun, 2006]. Chaque résultat correspond à une moyenne relative à cinq instances.

Taille	Type	Classe des coûts de mise en service	Moyenne des meilleures solutions connues
250×250	Symétrique	A	257 805,0
		B	276 035,2
		C	333 671,6
	Asymétrique	A	257 917,8
		B	276 053,2
		C	332 897,2
500×500	Symétrique	A	511 180,4
		B	537 912,0
		C	621 059,2
	Asymétrique	A	511 140,0
		B	537 847,6
		C	621 463,8
750×750	Symétrique	A	763 693,4
		B	796 571,8
		C	900 158,6
	Asymétrique	A	763 717,0
		B	796 374,4
		C	900 193,2

Table 7.2: Caractéristiques des instances du benchmark *Körkel-Ghosh* pour l'*UFLP*

7.6 Évaluations comparatives

Cette section présente les résultats obtenus avec notre méthode en comparaison avec ceux des différentes métaheuristiques détaillées dans la section 7.3 :

- Optimisation par essaims particulaires, *DPSO* (*Discrete Particle Swarm Optimization*, [Sevkli and Guner, 2006])
- Algorithme distribué de recuit simulé, *dESA* (*Distributed Evolutionary Simulated Annealing*, [Emin Aydin and Fogary, 2004])
- Recherche tabou, *TS* (*Tabu Search*, [Ghosh, 2003])

- Recherche locale complète avec mémoire, *CLM* (*Complete Local Search with Memory*, [Ghosh, 2003])
- Algorithme hybride, *Hybrid* ([Resende and Werneck, 2003])
- Recherche tabou avec mémoire à long terme, *TS-LTM* (*Tabu Search with Long Term Memory*, [Sun, 2006])

Pour comparer le temps d'exécution des différentes approches, nous estimons la performance des ordinateurs sur lesquels ont été réalisés les tests. La performance des matériels est estimée à partir des données recueillies dans [Dongarra, 2006]. Ces estimations de performances, reportées dans le tableau 7.3, permettent de normaliser les temps d'exécution relativement à un ordinateur de référence. Ici, l'ordinateur de référence est un *Pentium IV 3GHz* sur lequel notre approche a été exécutée. Nous rappelons que les temps convertis à l'aide de ces facteurs sont des approximations assez larges, mais néanmoins vérifiables. Nous avons pu vérifier le facteur de normalisation estimé pour l'approche *Hybrid* à partir d'une version proposée par Resende[4]. L'exécution sur notre *Pentium IV 3GHz* a confirmé le facteur estimé à 1/12.

Approche	Ordinateur	Performance estimée suivant [Dongarra, 2006] en Mflop/s	Facteur de normalisation du temps d'exécution
CBM	Pentium IV 3GHz	1571	1
DPSO	Pentium IV 2.6GHz	1190	1/1,32
dESA	Pentium III 700MHz 2 processeurs	514	1/3,06
TS	Celeron 650MHz	395	1/3,98
CLM	Celeron 650MHz	395	1/3,98
Hybrid	SGI Challenge 196MHz MIPS R10000	130	1/12,08
TS-LTM	Sun Enterprise 3000	110	1/14,28

Table 7.3: Facteurs de performance sur le benchmark *Linpack*, tiré de [Dongarra, 2006]

Dans les deux sections suivantes, nos résultats sont comparés à ceux des autres approches sur les benchmarks *OR-Library* et *Körkel-Ghosh*. Les résultats ont été obtenus avec le paramétrage donné dans le tableau 7.4 qui est le même pour le traitement de chaque instance. En comparaison du paramétrage utilisé pour le traitement du *VRP*, la taille de la coalition a été diminuée de manière à ce que les temps d'exécution soient comparables à ceux des autres approches traitant l'*UFLP*. Le protocole d'expérimentation reste identique à celui du chapitre précédent pour le *VRP*. La simulation est répétée 10 fois sur chaque instance et la valeur des résultats correspond à la moyenne sur ces 10 exécutions.

[4]Code source et exécutable de l'approche *Hybrid* de [Resende and Werneck, 2003] disponible sur : `http://www.research.att.com/~mgcr/popstar/`

Paramètre	Description	Valeur
C	Taille de la coalition	5
α	Poids initial	1,0
σ_1	Valeur de renforcement faible	0,2
σ_2	Valeur de renforcement forte	0,5
ρ	Coefficient d'imitation	0,3

Table 7.4: Paramétrages de *CBM* pour les benchmarks *OR-Library* et *Körkel-Ghosh*

7.6.1 Résultats sur le benchmark *OR-Library*

L'approche a été évaluée sur le benchmark *OR-Library* contenant des instances de petite et moyenne tailles. L'ensemble des approches étudiées fournissent une solution optimale en quelques secondes. La comparaison est donc effectuée sur les temps d'exécutions permettant d'obtenir ces valeurs de solutions. Le tableau 7.5, présente les temps moyens nécessaires à l'obtention des valeurs optimales pour *CBM* et pour quatre autres métaheuristiques pour lesquelles nous disposons des temps d'exécution : *DPSO* [Sevkli and Guner, 2006], *dESA* [Emin Aydin and Fogary, 2004], *Hybrid* [Resende and Werneck, 2003] et *TS-LTM* [Sun, 2006]. Les valeurs reportées dans le tableau correspondent aux temps normalisés, calculés à l'aide des facteurs de Dongarra donnés dans le tableau 7.3.

En comparant les différents résultats, on note que la différence de temps d'exécution s'apprécie sur les trois instances de plus grande taille "*capa*", "*capb*" et "*capc*". Les premières instances ont des tailles trop petites pour que nous puissions juger de l'efficacité réelle des approches. Sur ces 12 premières instances, seule l'approche *DPSO* semble être nettement la moins efficace.

En se basant sur les temps d'exécution moyens, les approches *DPSO* et *dESA* s'avèrent être les moins performantes. Pour ces deux approches, nous ne disposons pas d'autres résultats, notamment sur de plus grandes instances, permettant d'invalider ces résultats. L'approche *Hybrid* de Resende et Werneck produit les meilleurs temps d'exécution sur ce benchmark. Avec un temps d'exécution moyen de $0,33$ secondes, notre approche reste compétitive. La différence de temps avec les approches *Hybrid* et *TS-LTM*, avec des temps moyens respectifs de $0,11$ et $0,16$ secondes, pourrait être en partie imputée à l'imprécision portant sur le matériel.

7.6.2 Résultats sur le benchmark *Körkel-Ghosh*

La méthode a ensuite été testée sur le benchmark *Körkel-Ghosh* contenant des instances d'*UFLP* de grandes tailles. Sur ce benchmark, *CBM* est comparée à quatre autres approches : la méthode de recherche locale complète avec mémoire *CLM* [Ghosh, 2003], la recherche tabou de Ghosh *TS* [Ghosh, 2003], la méthode *Hybrid* [Resende and Werneck, 2003], et la recherche tabou de Sun *TS-LTM* [Sun, 2006]. Le tableau 7.6 présente

Instance	Taille	Optimum	Durées d'exécution normalisées (s)				
			CBM[a]	DPSO[b]	dESA[c]	Hybrid[d]	TS-LTM[e]
cap71	50x16	932615,75	0,00	0,01	0,00	0,00	0,00
cap72	50x16	977799,40	0,00	0,01	0,00	0,00	0,00
cap73	50x16	1010641,45	0,00	0,01	0,00	0,00	0,00
cap74	50x16	1034976,98	0,00	0,01	0,00	0,00	0,00
cap101	50x25	796648,44	0,00	0,06	0,02	0,01	0,00
cap102	50x25	854704,20	0,00	0,02	0,01	0,00	0,00
cap103	50x25	893782,11	0,01	0,05	0,04	0,01	0,00
cap104	50x25	928941,75	0,00	0,02	0,00	0,01	0,00
cap131	50x50	793439,56	0,02	0,43	0,07	0,01	0,01
cap132	50x50	851495,33	0,02	0,14	0,05	0,01	0,01
cap133	50x50	893076,71	0,02	0,32	0,03	0,01	0,01
cap134	50x50	928941,75	0,01	0,07	0,01	0,01	0,01
capa	1000x100	17156454,48	1,18	2,30	0,45	0,59	0,91
capb	1000x100	12979071,58	1,61	3,92	2,61	0,50	0,73
capc	1000x100	11505594,33	2,08	6,39	5,89	0,48	0,65
		Moyenne	0,33	0,92	0,61	0,11	0,16

[a] Durées sur un Pentium IV 3GHz
[b] Durées converties d'un Pentium IV 2.6GHz à un Pentium IV 3GHz
[c] Durées converties d'un Pentium III 700MHz 2proc. à un Pentium IV 3GHz
[d] Durées converties d'un SGI Challenge R10000 à un Pentium IV 3GHz
[e] Durées converties d'un Sun Enterprise 3000 à un Pentium IV 3GHz

Table 7.5: Résultats sur le benchmark *OR-Library*

les résultats correspondant à ces différentes approches. Dans les quatre premières colonnes du tableau, les caractéristiques des instances sont rappelées. La première colonne relative à *CBM* correspond au résultat moyen obtenu pour chaque classe d'instances. Étant donné que chaque ligne du tableau correspond à cinq instances d'une classe, chaque valeur reportée dans la colonne intitulée *Moyenne* est le résultat moyen de dix exécutions pour chacune des cinq instances. L'écart indiqué dans la seconde colonne de *CBM* correspond à l'écart relatif à la meilleure solution connue exprimé en pourcentage. Le temps d'exécution moyen en secondes est précisé dans la troisième colonne. Pour les quatre autres approches, sont donnés les écarts aux meilleures solutions connues ainsi que les temps moyens d'exécution. Ces temps d'exécution sont normalisés à l'aide des facteurs de Dongarra présentés dans le tableau 7.3.

Sur ce benchmark, les approches *CLM* et *TS* de Ghosh sont les moins performantes relativement au temps d'exécution et à la qualité des solutions. La durée d'exécution de ces deux approches est identique puisque dans le protocole d'expérimentation de Ghosh, la durée d'exécution de *CLM* est utilisée directement en critère d'arrêt pour *TS* [Ghosh, 2003].

Les approches *Hybrid* et *TS-LTM* sont ici les plus performantes. Cela confirme les résultats présentés pour le benchmark *OR-Library*. À noter que l'écart obtenu par *TS-LTM* est systématiquement de 0% étant donné que les meilleures valeurs connues en sont issues.

Bien que la méthode *TS-LTM* donne des résultats de qualité légèrement supérieure si on la compare avec l'approche *Hybrid*, la prise en compte conjointe de l'écart en termes de durée et de qualité de solution ne permet pas de les départager nettement.

Les performances de *CBM* sont globalement supérieures à celles de *CLM* et *TS*. En comparaison avec ces deux approches, les résultats sont de meilleure qualité avec des temps d'exécution comparables. Les durées d'exécution de notre approche sont plus importantes qu'avec les approches *Hybrid* et *TS-LTM*. La différence est d'un facteur 10. Les écarts aux meilleures solutions connues restent comparables. En moyenne, nous obtenons un écart de $0,009\%$ tandis que les approches *Hybrid* et *TS-LTM* obtiennent des écarts respectifs de $0,002\%$ et 0%.

L'efficacité de l'approche *Hybrid* peut être expliquée par l'utilisation d'une population de solutions et d'une procédure de *path-relinking*. Quant aux performances de *TS-LTM*, elles semblent liées à l'utilisation de plusieurs mémoires guidant la recherche et intervenant à différentes étapes. Ces éléments qui interviennent dans la performance, rendent les méthodes plus spécifiques au problème traité. Le choix réalisé dans *CBM* est de garder une certaine simplicité de spécialisation. Pour appliquer la méthode à l'*UFLP*, seuls des opérateurs d'intensification et de diversification nécessitaient d'être développées. De plus, par contraste à la plupart des approches considérées dans ce chapitre, *CBM* peut être implantée de manière distribuée sans qu'il ne soit nécessaire d'en modifier la structure.

7.7 Bilan

Nous avons présenté dans ce chapitre l'application de *CBM* sur un problème de positionnement discret sans contrainte de capacité. L'objectif est d'illustrer le caractère générique de notre approche. En séparant les opérateurs, des couches de décision et d'apprentissage, la spécialisation de *CBM* est facilitée. Pour traiter l'*UFLP*, seuls des opérateurs d'intensification et de diversification ont été développés.

CBM a été comparée à différentes approches heuristiques existantes. Sur les deux benchmarks considérés, les performances de *CBM* sont comparables à celles des approches les plus puissantes. De plus, notre métaheuristique intègre naturellement de nombreux avantages tels que des capacités d'adaptation par apprentissage, une certaine généricité et modularité obtenue par la délimitation précise des rôles et des composants à spécialiser, et la possibilité de distribuer l'exécution proprement dite.

Table 7.6: Résultats sur le benchmark *Körkel-Ghosh*

Problème				CBM			CLM		TS		Hybrid		TS-LTM	
Taille	Type		Meilleure	Moyenne	Écart	Durée (s)[a]	Écart	Durée (s)[b]	Écart	Durée (s)[b]	Écart	Durée (s)[c]	Écart	Durée (s)[d]
250	Symétrique	A	257805,0	257821,7	0,006%	3,7	0,035%	4,6	0,011%	4,6	0,001%	0,4	0,000%	0,2
		B	276035,2	276042,8	0,003%	3,2	0,115%	1,6	0,054%	1,6	0,000%	0,6	0,000%	0,4
		C	333671,6	333671,6	0,000%	4,2	0,000%	4,4	0,044%	4,4	0,000%	0,7	0,000%	0,7
	Asymétrique	A	257917,8	257938,0	0,008%	4,7	0,045%	4,5	0,023%	4,5	0,002%	0,4	0,000%	0,2
		B	276053,2	276063,8	0,004%	3,5	0,047%	1,6	0,150%	1,6	0,000%	0,6	0,000%	0,4
		C	332897,2	332913,4	0,005%	4,3	0,048%	6,2	0,104%	6,2	0,000%	0,6	0,000%	0,6
500	Symétrique	A	511180,4	511260,4	0,016%	32,0	0,060%	53,6	0,040%	53,6	0,003%	3,4	0,000%	1,1
		B	537912,0	538002,1	0,017%	22,8	0,144%	17,9	0,106%	17,9	0,001%	3,7	0,000%	2,2
		C	621059,2	621062,5	0,001%	29,5	0,018%	36,8	0,008%	36,8	0,000%	3,9	0,000%	5,0
	Asymétrique	A	511140,0	511175,6	0,007%	36,8	0,050%	52,0	0,022%	52,0	0,002%	2,8	0,000%	1,0
		B	537847,6	537916,7	0,013%	30,8	0,107%	19,9	0,055%	19,9	0,005%	3,5	0,000%	2,4
		C	621463,8	621538,9	0,012%	40,3	0,085%	33,8	0,067%	33,8	0,001%	4,5	0,000%	5,0
750	Symétrique	A	763693,4	763779,4	0,011%	105,9	0,037%	207,1	0,018%	207,1	0,002%	7,6	0,000%	2,8
		B	796571,8	796721,7	0,019%	72,3	0,076%	102,9	0,044%	102,9	0,004%	8,9	0,000%	6,5
		C	900158,6	900213,6	0,006%	95,5	0,070%	87,3	0,111%	87,3	0,004%	9,8	0,000%	16,1
	Asymétrique	A	763717,0	763801,7	0,011%	109,4	0,040%	211,9	0,016%	211,9	0,003%	8,1	0,000%	2,8
		B	796374,4	796471,8	0,012%	74,8	0,061%	99,5	0,061%	99,5	0,001%	9,5	0,000%	6,7
		C	900193,2	900224,9	0,004%	94,7	0,017%	125,6	0,036%	125,6	0,000%	10,4	0,000%	16,6
Moyenne					0,009%	42,7	0,059%	59,5	0,054%	59,5	0,002%	4,4	0,000%	3,9

[a] Durées sur un Pentium IV 3GHz
[b] Durées converties d'un Celeron 650MHz à un Pentium IV 3GHz
[c] Durées converties d'un SGI Challenge R10000 à un Pentium IV 3GHz
[d] Durées converties d'un Sun Enterprise 3000 à un Pentium IV 3GHz

QUATRIÈME PARTIE

Conclusion et perspectives

CHAPITRE 8

CONCLUSION

8.1 Bilan

Le travail de cette thèse a permis de fournir un ensemble d'outils pour l'analyse et la conception de métaheuristiques pour l'optimisation combinatoire en les formulant dans le cadre des systèmes multi-agents. D'une part, nous avons souhaité proposer des outils qui encouragent la modularité et la réutilisation des modèles. D'autre part, l'accent a été mis sur la potentialité de mise en œuvre distribuée des approches et sur l'utilisation de techniques d'apprentissage permettant de guider et d'adapter dynamiquement les méthodes de recherche. Dans cette optique, l'approche organisationnelle et l'approche agent ont été largement adoptées pour fournir un ensemble de concepts communs à différentes métaheuristiques. Ainsi, nous avons défendu l'idée que le paradigme des systèmes multi-agents peut être exploité avantageusement dans le domaine de l'optimisation. Cette section se propose d'examiner succinctement la démarche adoptée dans ce mémoire.

Cette thèse commence par dresser un état de l'art sur les métaheuristiques. Nous nous sommes particulièrement intéressés aux modèles et aux frameworks dont l'objectif principal vise à assister l'analyse et la conception de métaheuristiques. Il est ressorti de cette analyse, que si les modèles existants sont relativement bien adaptés à une phase d'analyse des métaheuristiques, ils ne semblent pas rendre compte suffisamment des aspects liés à la conception. En effet, ces approches n'intègrent pas les concepts de distribution de la résolution et de mise en œuvre distribuée. De même, les aspects d'adaptation et d'auto-adaptation de la recherche ne sont pas considérés. De plus, les approches ne fournissent que peu d'éléments méthodologiques pour guider pas à pas le concepteur dans la démarche de mise en œuvre. Cependant, plusieurs concepts intéressants sont introduits dans les différents modèles, et nous avons cherché à les exploiter dans le cadre de notre approche. Il s'agit principalement des concepts d'intensification, de diversification et de mémoire, que l'on peut identifier dans la plupart des métaheuristiques.

Suite à ce premier état de l'art, nous nous sommes penchés sur l'étude du rapport entre les métaheuristiques et les systèmes multi-agents. Nous avons tout d'abord présenté les méthodes collectives d'optimisation référencées dans la littérature comme étant des applications des systèmes multi-agents. Ensuite, nous avons observé les différents aspects des

métaheuristiques que l'on retrouve dans les systèmes multi-agents. L'analyse de ces différentes approches a permis d'identifier certains avantages liés à l'adoption du paradigme des systèmes multi-agents pour la conception de métaheuristiques. Tout d'abord, l'adoption d'une posture intentionnelle peut faciliter la description du fonctionnement des algorithmes en faisant référence à des agents autonomes poursuivants des buts. Ensuite, l'approche agent prend en compte en amont la possibilité de distribution de la résolution pour répartir l'exploration de l'espace de recherche entre différents processus ou pour mettre en concurrence différentes stratégies de recherche. De plus, l'exploitation de mécanismes issus de l'apprentissage artificiel dans les métaheuristiques apporte des solutions pour l'adaptation dynamique des stratégies de recherche. Enfin, l'approche agent bénéficie d'un ensemble d'outils de conception pouvant être adaptés aux métaheuristiques. Ces différents arguments en faveur des systèmes multi-agents nous confortent dans l'idée d'exploiter l'approche agent pour la conception de métaheuristiques.

En raison du manque d'outils pour la conception de métaheuristiques, nous avons proposé un framework organisationnel pour la modélisation et l'implantation de métaheuristiques. Ce framework est composé d'un modèle organisationnel auquel est associé un guide méthodologique. Le modèle adopte les concepts du méta-modèle RIO et présente ainsi une métaheuristique sous la forme d'une organisation composée de quatre rôles en interaction. Ce modèle peut être considéré comme un schéma de conception qui doit être raffiné pour obtenir un modèle de métaheuristique. Les règles méthodologiques proposent quelques étapes permettant de concevoir un système multi-agent à partir du modèle organisationnel initial. Le framework proposé donne un cadre d'analyse commun des différentes métaheuristiques et doit faciliter la conception de nouveaux algorithmes. L'usage de l'approche organisationnelle doit également assurer une bonne modularité du système et faciliter sa conception et sa mise en œuvre distribuée. Pour confirmer l'aspect générique des concepts introduits dans le framework, plusieurs métaheuristiques existantes ont été modélisées à l'aide de ses concepts.

Suite à cette première proposition, nous avons présenté une métaheuristique fondée sur la métaphore de la coalition, en utilisant le framework *AMF*. La coalition permet d'intégrer naturellement au système de résolution des aspects de distribution et de décentralisation du contrôle, de même que des procédés d'apprentissage individuels et collectifs. La recherche de solutions est effectuée par des agents possédant des caractéristiques identiques et jouant chacun l'ensemble des quatre rôles génériques du framework. Chaque agent est capable d'effectuer indépendamment des autres une recherche dans l'espace des solutions et d'adapter sa stratégie par apprentissage. De plus, un ensemble de mécanismes de coopération permet d'accroître la performance collective. Cette coopération entre les agents consiste, d'une part, à partager les meilleures solutions, et d'autre part, à échanger par mimétisme des informations sur les stratégies de recherche performantes.

Nous avons ainsi introduit dans la métaheuristique des caractéristiques de plusieurs

autres méthodes. Notamment, nous reprenons un principe des essaims particulaires par la prise en compte simultanée de la meilleure solution partagée connue, de la meilleure solution trouvée et de la solution courante. De même, nous retrouvons des traits communs aux algorithmes mémétiques du fait des recherches locales indépendantes réalisées. Enfin, l'apprentissage des bons choix d'opérateurs reprend le principe des hyper-heuristiques, selon des règles simplifiées de mise à jour des pondérations condition/action, et tout en étendant leur principe vers un contexte de résolution collective. Nous pouvons dire que le système multi-agent réalise une hybridation de différents mécanismes de métaheuristiques, sous couvert d'une métaphore simple de coalition. Enfin, notons que les mécanismes d'auto-adaptation de la recherche sont ici clairement identifiés et localisés à un niveau indépendant du problème spécifique à traiter.

Pour valider la métaheuristique à base de coalition, nous avons traité deux problèmes d'optimisation : un problème de tournées de véhicules (*VRP*), et un problème de positionnement (*UFLP*). Pour ces deux applications, nous avons détaillé dans un premier temps la démarche de spécialisation qui consiste essentiellement en la définition des opérateurs de bas niveau d'intensification et de diversification. Ceux-ci sont implantés sous une forme standard. Ensuite, nous avons analysé l'impact des principaux composants de la métaheuristique sur ses performances. Ainsi, a été mis en évidence un apport positif des mécanismes d'apprentissage et de coopération. Enfin, les résultats ont été comparés avec ceux obtenus avec plusieurs autres méthodes heuristiques existantes considérées dans la littérature comme étant parmi les plus puissantes sur ces problèmes. Cette évaluation comparative réalisée en termes de qualité des solutions et de temps d'exécution semble indiquer que la métaheuristique à base de coalition est effectivement compétitive vis-à-vis des approches existantes.

Ces applications ont permis d'illustrer concrètement une démarche de conception de métaheuristiques fondée sur le paradigme agent. Toutefois, le cadre global que nous cherchons à construire avec ce travail n'est pas encore achevé. Divers points requièrent d'avantages d'analyses et d'approfondissements. Ils sont détaillés dans la section suivante.

8.2 Perspectives et travaux futurs

Nous définissons deux axes d'approfondissement pour la suite de ce travail. À court terme, le premier point concerne l'amélioration de la métaheuristique à base de coalition, notamment pour approfondir les questions de mise en œuvre distribuée et pour confronter le modèle de décision et d'apprentissage à d'autres modèles. Le second axe porte sur le framework proposé et concerne, à moyen et long termes, l'élaboration d'une méthodologie plus complète et mieux formalisée.

8.2.1 Exécution distribuée de CBM

Compte tenu des résultats prometteurs de la métaheuristique à base de coalition, il semble intéressant de pouvoir tester son exécution sur un réseau d'ordinateurs. Les développements nécessaires à cette mise en œuvre distribuée peuvent être facilités par l'utilisation d'une plateforme multi-agent. Cette dernière pourra être choisie en fonction de sa capacité à fournir les éléments nécessaires à l'implantation de modèles organisationnels, et avant tout, en fonction des possibilités de distribution de l'exécution. Plusieurs plateformes prennent en compte ces aspects, par exemple MadKit[1], ou encore Janus[2].

Cette distribution nécessite d'autre part d'approfondir l'étude de notre approche pour étudier la complexité en termes de messages échangés entre les agents. Dans la métaheuristique à base de coalition, les agents communiquent dans un réseau complet et d'autres topologies de réseaux moins coûteuses pourraient être envisagées.

8.2.2 De nouveaux modèles de décision et d'apprentissage

Dans la métaheuristique à base de coalition, un grand nombre de possibilités de choix dans les règles d'apprentissage peut être envisagé, aussi bien pour le mécanisme d'apprentissage par renforcement, que pour le mimétisme. Il serait intéressant de développer et de comparer d'autres règles de mise à jour du modèle de sélection d'actions, voire de les combiner, afin d'améliorer les performances globales de la méthode. Ces améliorations peuvent porter, dans un premier temps, sur les conditions de déclenchement du renforcement ou du mimétisme, ou encore sur le calcul des modifications de pondérations des couples condition/opérateur. D'autre part, il serait intéressant de prendre en compte la durée d'exécution des opérateurs, en plus du gain réalisé sur le coût de la solution, pour évaluer l'efficacité des opérateurs. Le positionnement de la méthode en tant qu'extension des approches hyper-heuristiques dans un contexte de résolution collective mérite d'être mieux pris en compte.

D'autres mécanismes classiques de sélection d'opérateurs et d'apprentissage peuvent être transposés et appliqués sur l'architecture de la coalition. Par exemple, la sélection d'opérateurs pourrait être réalisée par des réseaux neuronaux à condition d'adapter et de transposer leurs mécanismes d'apprentissage dans le contexte non supervisé qui est le notre. Car ici l'agent ne dispose que du retour d'expérience au cours de la recherche.

8.2.3 Vers une méthodologie pour la conception de métaheuristiques

La dernière perspective de ce travail porte sur l'élaboration d'une méthodologie plus complète pour la conception de métaheuristiques. Le framework proposé est constitué d'un modèle organisationnel et d'un guide méthodologique qui nécessitent d'être complétées. Pour cela, il semble nécessaire de mieux formaliser le processus de développement. Il

[1] MadKit : `http://www.madkit.org/`
[2] Janus : `http://www.janus-project.org/`

semble intéressant de pouvoir associer des modèles ou méta-modèles à chacune des étapes du processus de développement, puis d'élaborer des outils logiciels d'assistance au développement et de génération de code. Ces modèles doivent intégrer des éléments plus généraux sur les systèmes multi-agents comme les concepts d'"agent" ou de "groupe", mais aussi intégrer le domaine de l'optimisation avec des concepts tels que "Problème d'optimisation", "Solution" ou "Structure de voisinage".

L'élaboration de cette méthodologie peut être facilitée en intégrant le framework proposé dans une méthodologie déjà existante pour la conception de systèmes multi-agent. Étant donné que nous avons déjà utilisé des éléments du méta-modèle RIO, il semble intéressant de pouvoir continuer dans cette voie en exploitant CRIO une version étendue du méta-modèle RIO [Gaud, 2007]. L'intérêt de ce méta-modèle est qu'il est associé à un ensemble d'outils logiciels d'aide à la conception et au développement. Cette intégration permettrait de reformuler notre démarche dans le cadre de la méthodologie générale ASPECS, dans laquelle s'insère déjà CRIO, et éventuellement de l'adapter aux problématiques particulières soulevées dans les métaheuristiques.

Bibliographie

Alba, E. and Tomassini, M. (2002). Parallelism and evolutionary algorithms. *IEEE Transaction on Evolutionary Computation*, 6(5) :443–462.

Bachem, A., Hochstattler, W., and Malich, M. (1994). Simulated trading a new parallel approach for solving vehicle routing problems. In *International Conference on Parallel Computing : Trends and Applications*.

Bäck, T., Eschelman, L. J., Rudolph, G., Porto, V. W., Kinnear, K. E., Smith, R. E., and Michalewicz, Z. (1997). *Handbook of evolutionary computation*, chapter B1. Evolutionary algorithms and their standard instances. CRC Press.

Battiti, R. (1996). *Modern Heuristic Search Methods*, chapter Reactive Search : Toward Self-Tuning Heuristics, pages 61–83. John Wiley and Sons Ltd.

Battiti, R. and Protasi, M. (1996). Reactive search, a history-based heuristic for max-sat. In *Workshop on Satisfiability*.

Battiti, R. and Tecchiolli, G. (1994). The reactive tabu search. *INFORMS Journal on Computing*, 6(2) :126–140.

Beasley, J. E. (1993). Lagrangean heuristics for location problems. *European Journal of Operational Research*, 65 :383–399.

Birattari, M. (2005). *The Problem of Tuning Metaheuristics*. PhD thesis, Université Libre de Bruxelles.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Surveys*, 35(3) :268–308.

Branke, J., Stein, M., and Schmeck, H. (2005). *Handbook of Bioinspired Algorithms and Applications*, chapter A unified view on metaheuristics and their hybridization, pages 147–156. CRC Press.

Burke, E., Hart, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003). *Handbook of Meta-Heuristics*, chapter Hyper-Heuristics : An Emerging Direction in Modern Search Technology, pages 457–474. Kluwer Academic.

Bäck, T., Rudolph, G., and Schwefel, H.-P. (1993). Evolutionary programming and evolution strategies : Similarities and differences. In *Conference on Evolutionary Programming*, pages 11–22.

Bürckert, H. J., Fischer, K., and Vierke, G. (1998). Transportation scheduling with holonic mas - the teletruck approach. In *International Conference on Practical Applications of Intelligent Agents and Multiagents, (PAAM'98)*, pages 577–590.

Cahon, S., Melab, N., and Talbi, E.-G. (2004). Paradiseo : A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10 :357–380.

Christofides, N., Mingozzi, A., and Toth, P. (1979). *Combinatorial Optimization*, chapter The vehicle routing problem, pages 315–338. Wiley.

Chrétien-Goni, J.-P. (2005). Heuristique. *Encyclopædia Universalis*, [En ligne].

Clair, G., Kaddoum, E., Gleizes, M.-P., and Picard, G. (2008). Approches multi-agents auto-organisatrices pour un contrôle manufacturier intelligent et adaptatif. In Cépaduès, editor, *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2008)*, pages 191–200.

Clerc, M. (2005). *L'optimisation par essaims particulaires : versions paramétriques et adaptatives*. Lavoisier, Hermes Science Publications.

Collette, Y. and Siarry, P. (2002). *Optimisation multiobjectif*. Eyrolles. ISBN : 2212111681.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.-S. (2004). New heuristics for the vehicle routing problem. In *Les Cahiers du GERAD*.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5) :512–522.

Cornuéjols, G., Nemhauser, G. L., and Wolsey, L. A. (1990). *Discrete Location Theory*, chapter The uncapacitated facility location problem, pages 199–171. John Wiley and Sons, New York.

Cowling, P., Kendall, G., and Soubeiga, E. (2001). A parameter-free hyperheuristic for scheduling a sales summit. In *Metaheuristics International Conference, MIC 2001*, pages 127–131.

Cowling, P. I., Kendall, G., and Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. In *International Conference on Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes In Computer Science*, pages 176–190.

Crainic, T. and Toulouse, M. (2003). *State-of-the-Art Handbook in Metaheuristics*, chapter Parallel Strategies for Meta-heuristics, pages 475–513. Kluwer Academic Publishers.

Crainic, T. G. and Semet, F. (2006). *Optimisation combinatoire 3 - applications*, chapter Recherche opérationnelle et transport de marchandises, pages 47–115. Lavoisier, Hermes Science.

Créput, J.-C. and Koukam, A. (2008). The memetic self-organizing map approach to the vehicle routing problem. *Softcomputing applications in industry*, 12 :189–205.

Current, J., Daskin, M., and Schilling, D. (2004). *Facility location : Application and theory*, chapter Discrete network location models, pages 81–118. Springer-Verlag.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1) :80–91.

Devarenne, I. (2007). *Études en recherche locale adaptative pour l'optimisation combinatoire*. PhD thesis, Université de Technologie de Belfort Montbéliard.

Dongarra, J. J. (2006). Performance of various computers using standard linear equations software. Technical Report CS-89-85, Computer Science Department, University of Tennessee and Computer Science and Mathematics Division, Oak Ridge National Laboratory.

Dorigo, M., Caro, G. D., and Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial life*, 5(2) :137–172.

Dorigo, M. and Stützle, T. (2000). The ant colony optimization metaheuristic : Algorithms, applications and advances. Technical Report IRIDIA-2000-32, IRIDIA.

Dorne, R. and Voudouris, C. (2004). *Metaheuristics : computer decision-making*, chapter HSF : the iOpt's framework to easily design metaheuristic methods, pages 237–256. Kluwer Academic.

Drogoul, A. (2005). *L'intelligence (Traité des Sciences cognitives)*, chapter Les systèmes multi-agents. Hermes Science.

Dréo, J., Aumasson, J.-P., Tfaili, W., and Siarry, P. (2007). Adaptive learning search, a new tool to help comprehending metaheuristics. *International Journal on Artificial Intelligence Tools*, 16 :483–505.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2) :124–141.

Eiben, A. E. and Schippers, C. A. (1998). On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35 :1–16.

Emin Aydin, M. and Fogary, T., C. (2004). A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. *Journal of Heuristics*, 10 :269–292.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6) :992–1009.

Eshelman, L. J. (1997). *Handbook of Evolutionary Computation*, chapter Genetic algorithms. IOP Publishing Ltd and Oxford University Press.

Ferber, J. (1995). *Les systèmes multi-agents, vers une intelligence collective*. Intereditions edition.

Ferber, J. and Jacopin, E. (1991). The framework of eco problem solving. *Decentralized Artificial Intelligence*, 2.

Fink, A. and Voss, S. (2002). *Optimization Software Class Libraries*, chapter Hotframe : A Heuristic Optimization Framework, pages 81–154.

Förster, M., Bickel, B., Bernd, H., and Kókai, G. (2007). Self-adaptive ant colony optimisation applied to function allocation in vehicle networks. In *Genetic and Evolutionary Computation Conference, GECCO*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1999). *Design Patterns, Catalogue de modèles de conceptions réutilisables*. Vuibert.

Gaud, N. (2007). *Systèmes multi-agents holoniques : de l'analyse à l'implantation. Méta-modèle, méthodologie, et simulation multi-niveaux*. PhD thesis, Université de Franche-Comté et Université de Technologie de Belfort-Montbéliard.

Gendreau, M. (2003). *Handbook of Metaheuristics*, chapter An introduction to tabu search, pages 37–54. Kluwer Academic.

Ghosh, D. (2003). Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research*, 150 :150–162.

Ghédira, K. (1994). Distributed simulated re-annealing for dynamic constraint satisfaction problems. In *International Conference on Tools with Artificial Intelligence*, pages 601–607.

Glover, F. (1997). Tabu search and adaptive memory programming : advances, applications and challenges. *Advances in metaheuristics, optimization and stochastic modeling technologies*, pages 1–75.

Glover, F. and Kochenberger, G. A. (2003). *Handbook of Metaheuristics*. Kluwer Academic.

Glover, F., Laguna, M., and Marti, R. (2003). *Handbook of Metaheuristics*, chapter Scatter search and path relinking : advances and applications, pages 1–36. Kluwer Academic.

Goldberg, D. E. (1994). *Algorithmes génétiques*. Addison-Wesley France.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998). *Fleet management and logistics*, chapter The impact of metaheuristics on solving the vehicle routing problem : algorithms, problem sets, and computational results, pages 33–56. Kluwer Academic.

Gruer, P., Hilaire, V., Koukam, A., and Cetnarowicz, K. (2002). A formal framework for multi-agent systems analysis and design. *Expert Systems with Applications*, 23(4) :349–355.

Guner, A. R. and Sevkli, M. (2008). A discrete particle swarm optimization algorithm for uncapacitated facility location problem. *Journal of Artificial Evolution and Applications*, 2008 :9.

Hamacher, H. W., Nickel, S., and Schneider, A. (1996). Classification of location problems. Technical report, Fachbereich Mathematik Universität Kaiserslautern.

Hansen, P. and Mladenović, N. (2003). *Handbook of Metaheuristics*, chapter Variable Neighborhood Search, pages 145–184. Kluwer Academic.

Henderson, D., Jacobson, S. H., and Johnson, A. W. (2003). *Handbook of Metaheuristics*, chapter The theory and practice of simulated annealing, pages 287–320. Kluwer Academic.

Hilaire, V. (2000). *Vers une approche de spécification, de prototypage et de vérification de systèmes multi-agents*. PhD thesis, Université de Franche-Comté.

Hinterding, R., Michalewicz, Z., and Eiben, A. E. (1997). Adaptation in evolutionary computation : a survey. In *IEEE International Conference on Evolutionary Computation*, pages 65–69.

Hinterding, R., Michalewicz, Z., and Peachey, T. C. (1996). Self-adaptive genetic algorithm for numeric functions. In *Parallel Problem Solving from Nature, PPSN IV*, Lecture Notes in Computer Science.

Hoffmeister, F. and Bäck, T. (1991). Genetic algorithms and evolution strategies : Similarities and differences. In *Parallel Problem Solving from Nature*, pages 455–469.

Holland, J. H., Booker, L. B., Colombetti, M., Dorigo, M., Goldberg, D. E., Forrest, S., Riolo, R. L., Smith, R. E., Lanzi, P. L., Stolzmann, W., and Wilson, S. W. (2000). What is a learning classifier system ? *Lecture Notes in Computer Science*, 1813 :3–32.

Horling, B. and Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19 :281–316.

Hu, B. and Raidl, G. R. (2006). Variable neighborhood descent with self-adaptive neighborhood-ordering. In *7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics*.

Ingber, L. (1996). Adaptive simulated annealing (asa) : Lessons learned. *Journal of Control and Cybernetics*, 25(1) :33–54.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *International Journal of Autonomous Agents and Multi-Agent Systems*, 1(1) :7–38.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948.

Kennedy, J. and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4104–4108.

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

Kohonen, T. (2001). *Self-Organizing Maps*. Springer, Berlin, third edition.

Körkel, M. (1989). On the exact solution of large-scale simple plant location problems. *European Journal of Operational Research*, 39(2) :157–173.

Kozlak, J., Créput, J., Hilaire, V., and Koukam, A. (2004). Multi-agent environment for dynamic transport planning an scheduling. In *Lecture Notes in Computer Science*, volume 3038, pages 638–645. Springer-Verlag.

Krasnogor, N. and Smith, J. (2005). A tutorial for competent memetic algorithms : Model, taxonomy, and design issues. *IEEE Transaction on Evolutionary Computation*, 9(5) :474–488.

Kuehn, A. A. and Hamburger, M. J. (1963). A heuristic program for locating warehouse. *Mangement Science*, 9(9) :643–666.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4-5) :285–300.

Lenstra, J. and Rinnooy Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2) :221–227.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44 :2245–2269.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). *Handbook of Metaheuristics*, chapter Iterated Local Search, pages 321–353. Kluwer Academic.

Maniezzo, V. and Carbonaro, A. (2001). *Essays and Surveys in Metaheuristics*, chapter Ant Colony Optimization : an Overview, pages 21–44. Kluwer Academic.

Meignan, D., Créput, J.-C., and Koukam, A. (2008a). A coalition-based metaheuristic for the vehicle routing problem. In *IEEE Congress on Evolutionary Computation*, pages 1176–1182.

Meignan, D., Créput, J.-C., and Koukam, A. (2008b). An organizational view of metaheuristics. In N. R. Jennings, A. Rogers, A. P. and Ramchurn, S. D., editors, *First International Workshop on Optimisation in Multi-Agent Systems, AAMAS'08*, pages 77–85.

Meignan, D., Créput, J.-C., and Koukam, A. (2008c). Coalition-based metaheuristic : A self-adaptive metaheuristic using reinforcement learning and mimetism. In *Workshop on Hyper-heuristics, International Conference on Parallel Problem Solving from Nature (PPSN)*.

Merz, P. (2000). *Memetic Algorithms for Combinatorial Optimization Problems : Fitness Landscapes and Effective Search Strategies*. PhD thesis, Universität-Gesamthochschule Siegem.

Mester, D. and Bräysy, O. (2005). Active guided evolution strategies for large scale vehicle routing problems with time windows. *Computers & Operations Research*, 32 :1593–1314.

Mester, D. and Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34(10) :2964–2975.

Milano, M. and Roli, A. (2004). Magma : a multiagent architecture for metaheuristics. *IEEE Transaction on Systems, Man, and Cybernetics, Part B*, 34(2) :925–941.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. Technical report, Caltech Concurrent Computation Program.

Moujahed, S. (2007). *Approche multi-agents auto-organisée pour la résolution de contraintes spaciales dans les problèmes de positionnement mono et multi-niveaux*. PhD thesis, Université de Franche-Comté, Université de Technologie de Belfort-Montbéliard.

Moujahed, S., Gaud, N., and Meignan, D. (2007). A self-organizing and holonic model for optimization in multi-level location problems. In *IEEE International Conference on Industrial Informatics, INDIN 2007*.

Moujahed, S., Simonin, O., Koukam, A., and Ghédira, K. (2006). Self-organizing multiagent approach to optimization in positioning problems. In *European Conference on Artificial Intelligence*, pages 275–279.

Naddef, D. and Rinaldi, G. (2001). *The Vehicle Routing Problem*, chapter Branch-and-cut algorithms for the capacitated VRP, pages 53–84. SIAM.

Oliver, I. M., Smith, D. J., and Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. In Grefenstette, J. J., editor, *International Conference on Genetic Algorithms*, pages 224–230.

Ong, Y.-S., Lim, M.-H., Zhu, N., and Wong, K.-W. (2006). Classification of adaptive memetic algorithms : a comparative study. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 36(1) :141– 152.

Özcan, E., Bilgin, B., and Korkmaz, E. E. (2006). Hill climbers and mutational heuristics in hyperheuristics. In *Parallel Problem Solving from Nature, PPSN IX*, pages 202–211.

Özcan, E., Bilgin, B., and Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1) :3–23.

Parunak, H. V. D., Brueckner, S., Fleischer, M., and Odell, J. (2003). A design taxonomy of multi-agent interactions. *Lecture Notes in Computer Science*, 2935(4) :123–137.

Picard, G., Gleizes, M.-P., and Glize, P. (2007). Distributed frequency assignment using cooperative self-organization. In *International Conference on Self-Adaptive and Self-Organizing Systems*, pages 183–192.

Piechowiak, S. and Hamadi, Y. (2002). *Organisation et applications des SMA*, chapter Problèmes de satisfaction de contraintes et systèmes multi-agents, pages 169–205. Hermès Science Publications, Paris.

Pilat, M. L. and White, T. (2002). Using genetic algorithms to optimize acs-tsp. In *Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 101–172. Springer Berlin, Heidelberg.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31 :1985–2002.

Randall, M. (2004). Near parameter free ant colony optimisation. In *Ant Colony, Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 374–381. Springer Berlin, Heidelberg.

Rao, A. S. and Georgeff, M. P. (1995). Bdi agents : From theory to practice. Technical report, Australian Artificial Intelligence Institute.

Resende, M. G. C. and Werneck, R. F. (2003). A hybrid multistart heuristic for the uncapacitated facility location problem. Technical Report TD-5RELRR, AT&T Labs.

Roli, A. and Milano, M. (2002). Magma : A multiagent architecture for metaheuristics. Technical Report DEIS-LIA-02-007, Universita degli Studi di Bologna.

Ropke, S. and Pisinger, D. (2005). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Technical report, University of Copenhagen.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40 :421–438.

Sakarovitch, M. (1984). *Optimisation combinatoire, Méthodes mathématiques et algorithmiques - Programmation discrète*. Hermann. ISBN : 2-7056-5976-5.

Sevkli, M. and Guner, A. R. (2006). A continuous particle swarm optimization algorithm for uncapacitated facility location problem. In *Ant colony optimization and swarm intelligence*.

Shaefer, C. G. (1987). The argot strategy : adaptive representation genetic optimizer technique. In *Second International Conference on Genetic Algorithms and their application*.

Skolicki, Z. (2005). An analysis of island models in evolutionary computation. In *Genetic And Evolutionary Computation Conference*, pages 386–389.

Smith, R. G. (1980). The contract net protocol : high-level communication and control in a distributed problem solver. *IEEE Transactions on Computer*, 29 :1104–1113.

Sun, M. (2006). Solving the uncapacitated facility location problem using tabu search. *Computers & Operations Research*, 33(9) :2563–2589.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning : Introduction. Technical report, Cognitive Science Research Group.

Taillard, E. D., Gambardella, L. M., Gendreau, M., and Potvin, J.-Y. (2001). Adaptive memory programming : A unified view of metaheuristics. *European Journal of Operational Research*, 135 :1–16.

Talbi, E.-G. and Bachelet, V. (2004). Cosearch : A parallel co-evolutionary metaheuristic. In *Int. workshop on hybrid metaheuritics*, pages 127–140.

Tanese, R. (1989). Distributed genetic algorithms. In *International Conference on Genetic Algorithms*, pages 434–439.

Toth, P. and Vigo, D. (2001). *The Vehicle Routing Problem*, chapter Branch-and-bound algorithms for the capacitated VRP, pages 29–51. SIAM.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing*, 15 :333–348.

Tsang, E. P. K. (1993). *Foundations of Constraint Satisfaction*. ISBN : 0127016104.

Voudouris, C. and Tsang, E. P. K. (2003). *Handbook of Metaheuristics*, chapter Guided local search, pages 185–218. Kluwer Academic.

Wooldridge, M. (1992). *The logical modelling of computational multi-agent systems*. PhD thesis, University of Manchester.

Wooldridge, M. and Jennings, N. (1995). Intelligent agents : Theory and practice. *Knowledge Engineering Review*, 10(2).

Wren, A. and Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 23(3) :333–344.

Yamaguchi, T., Tanaka, Y., and Yachida, M. (1997). Speed up reinforcement learning between two agents with adaptive mimetism. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 594–600.

Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. (1998). The distributed constraint satisfaction problem : Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5) :673–685.

Zlochin, M., Birattari, M., Meuleau, N., and Dorigo, M. (2004). Model-based search for combinatorial optimization : A critical survey. *Annals of Operations Research*, 131 :373–395.