Programmation à mémoire adaptative et algorithmes pseudo-gloutons : nouvelles perspectives pour les méta-heuristiques

Thèse présentée à

L'Université de Versailles Saint-Quentin-en-Yvelines

pour l'obtention de
l'habilitation à diriger des recherches

par

Éric Denis Taillard
Docteur ès Sciences, Ingénieur informaticien
Né à La Chaux-de-Fonds, Suisse

Acceptée sur proposition du jury

Professeur	Catherine Roucairol, directrice
Professeur	Jacques Teghem, président
Professeur	Michel Gendreau, rapporteur
Professeur	Fred Glover, rapporteur
Professeur	Gérard Plateau, rapporteur
Professeur	Nelson Maculan, membre

Versailles, le 4 décembre 1998

RÉSUMÉ

La première partie de ce dossier analyse les développements récents de plusieurs méta-heuristiques à mémoire et montre que les manières d'implanter l'une ou l'autre de ces méthodes générales d'optimisation ont tendance à se rapprocher. Une présentation unifiée de ces méthodes est proposée sous le nom de programmation à mémoire adaptative. *Un certain nombre de méthodes récemment développées pour la résolution de problèmes d'affectation quadratique et d'élaboration de tournées de véhicules sont passées en revue et présentées sous l'angle de la programmation à mémoire adaptative.*

Une seconde partie est consacrée à un des mécanismes fondamentaux de la programmation à mémoire adaptative, à savoir une procédure de recherche locale. Dans certains cas, une méthode d'optimisation gloutonne s'arrêtant au premier optimum local relativement à un voisinage simple convient, mais il est parfois nécessaire d'utiliser une procédure plus élaborée. On peut penser à implanter une méthode évoluée, comme par exemple une recherche avec tabous, mais il n'est parfois pas nécessaire d'aller jusque là : Dans bien des cas en effet, l'usage d'un second voisinage ou d'une fonction-objectif alternative permet d'améliorer suffisamment un algorithme glouton ; on parlera alors de méthode pseudo-gloutonne.

Finalement, on montre comment un programme à mémoire adaptative ou une recherche pseudo-gloutonne peuvent se paralléliser de façon naturelle et efficace.

TABLE DES MATIÈRES

1. INTRODUCTION

Nos travaux ont porté dans plusieurs directions. Ils ont été publiés, ou soumis pour publication en ce qui concerne les plus récents, dans des revues de niveau international avec comité de lecture. Par conséquent, nous ne ferons qu'une synthèse sommaire de nos activités scientifiques, renvoyant le lecteur aux diverses publications jointes en annexe s'il désire des informations plus détaillées et plus complètes. Nous avons divisé notre activité scientifique en trois lignes directrices, chacune présentée dans un chapitre distinct. Nous commencerons par la synthèse de nos travaux réalisés dans le cadre des méthodes générale d'optimisation approchées, ou méta-heuristiques. Nous avons intitulé ce chapitre *programmation à mémoire adaptative* car il nous semble que cette appellation qualifie fidèlement les évolutions les plus récentes de plusieurs méta-heuristiques que nous avons étudiées. Tout d'abord, nous passerons en revue un certain nombre de méta-heuristiques dotées d'une mémoire, comme les algorithmes génétiques, la recherche par dispersion, la recherche avec tabous et les colonies de fourmis. Bien que ces méthodes ont peu de choses en commun a priori, nous montrerons que diverses de leurs implantations récentes sont basées en fait sur un ensemble restreint de principes, ce qui justifiera la synthèse très concise que nous avons faite sous l'appellation de programmation à mémoire adaptative.

Cette dernière présente plusieurs avantages. Premièrement, elle produit des solutions de qualité très élevée, ce qui nous a permis d'améliorer notablement maints problèmes d'optimisation combinatoire. Ensuite, elle se prête bien au traitement de problèmes dynamiques. Enfin, elle est facilement adaptable au calcul parallèle.

Une des deux composantes principales d'un programme à mémoire adaptative est une procédure de recherche locale. Le deuxième chapitre discutera précisément d'un nouveau type de recherche locale, que nous avons appelé *algorithme pseudo-glouton*, et qui fait appel à des voisinages combinés. En effet, nous avons pu obtenir de très bons résultats à l'aide de méthodes pseudo-gloutonnes. Pour certains problèmes, comme celui de la p-médiane, il est même possible de se passer de la procédure à mémoire adaptative, tant la qualité des solutions est élevée.

Le troisième chapitre sera consacré à la parallélisation des méthodes à mémoire adaptative et pseudo-gloutonnes. Cette voie est intéressante à plus d'un égard : d'abord, la conception d'algorithmes parallèles doit se faire en adoptant un point de vue très éloigné de celui d'une approche séquentielle, ce qui favorise le développement d'algorithmes fonctionnant sur des bases radicalement différentes, difficilement imaginables dans un monde séquentiel. Ensuite, les programmes à mémoire adaptative requièrent parfois des temps de calcul importants, incompatibles avec un traitement en temps réel par une machine séquentielle. Certains de nos programmes à mémoire adaptative ont justement été conçus dans une philosophie de calcul concurrent et ont été implantés sur machine distribuée.

2. La programmation à mémoire adaptative

2.1. Introduction

Les méthodes génériques et heuristiques d'optimisation, appelées aussi méta-heuristiques ou méthodes de recherche locale générales, se développent de manière explosive à l'heure actuelle : toutes les plus importantes conférences en recherche opérationnelle ont au moins une session, sinon un groupe de sessions, entièrement consacrés aux méta-heuristiques ; la littérature leur fait également une place de plus en plus grande. Parmi les techniques qui connaissent le plus de succès, on peut citer par exemple les algorithmes génétiques, le recuit simulé, la recherche avec tabous, la recherche par dispersion, les systèmes de fourmis, sans parler d'autres « nouvelles techniques » qui fleurissent chaque jour. Le succès de ces méthodes est dû à plusieurs facteurs, comme leur facilité d'implantation, leur capacité à prendre en considération avec flexibilité des contraintes spécifiques apparaissant dans les applications pratiques ainsi que la qualité élevée des solutions qu'elles sont capables de trouver.

D'un point de vue théorique, l'usage de méta-heuristiques n'est pas vraiment justifié et les résultats obtenus sont plutôt faibles. Par exemple, il existe quelques théorèmes de convergence pour le recuit simulé [HAJ88] ou la recherche avec tabous [FAI92] qui sont inutilisables en pratique : ils établissent seulement que l'on a une probabilité très élevée de trouver une solution globalement optimale, si un temps de calcul complètement disproportionné est alloué, temps supérieur à celui nécessaire pour énumérer complètement l'espace de toutes les solutions.

Cependant, en pratique, les performances de ces techniques sont excellentes et elles sont souvent extrêmement compétitives. Dans cette course à l'excellence, on observe que les méthodes les plus efficaces hybrident deux ou plusieurs méta-heuristiques. Il en résulte que des techniques ayant à peu près les mêmes principes de fonctionnement portent des noms très différents comme « hybride génétique », « recherche avec tabous probabiliste », « système de fourmis MAX-MIN ». Toutes ces méthodes partagent trois particularités : premièrement elles mémorisent des solutions ou des caractéristiques des solutions visitées durant le processus de recherche ; deuxièmement, elles font usage d'une procédure créant une nouvelle solution à partir des informations mémorisées et troisièmement, elles sont dotées d'une recherche locale, que ce soit une méthode gloutonne, une recherche avec tabous élémentaire ou un recuit simulé.

Considérant la grande similitude existant entre ces méthodes, une présentation unifiée se justifie, de même qu'un nom à la fois plus synthétique et plus général. Ceci permet également d'échapper aux idiotismes d'une école professant l'usage de termes spécifiques de moins en moins en relation avec le fonctionnement réel de la méthode. Nous proposons de regrouper sous un même toit ces méta-heuristiques et de les qualifier de programmation à mémoire adaptative. Ces termes ont déjà été proposés par Glover [GLO97b] dans le contexte de la recherche avec tabous. En effet, cette dernière a toujours été présentée comme étant dotée d'au moins une mémoire (la liste de tabous) et ouverte à toutes les composantes de l'intelligence artificielle. Les évolutions successives de l'implantation d'une recherche avec tabous pourraient cependant facilement mener à une méthode sans liste de tabous (mais avec une autre forme de mémoire). Le nom de la technique devient par là difficile à justifier sinon pour des raisons historiques. Ainsi, le souci de Glover dans [GLO97c] est de mieux qualifier une telle méthode sans tabous, basée sur d'autres ingrédients déjà proposés dans ses articles de base [GLO89, GLO90].

Un phénomène similaire peut se poser pour d'autres méta-heuristiques. Par exemple, des applications d'algorithmes génétiques ne codent plus une solution du problème sous la forme d'un vecteur binaire et les opérateurs de croisement et de mutation n'ont souvent plus rien à voir avec les opérateurs standard pour lesquels quelques résultats théoriques ont été établis. Nous allons donc commencer par passer une revue et commenter les principales méta-heuristiques à mémoire avant d'en faire une synthèse et de les présenter sous l'angle de la programmation à mémoire adaptative. Ensuite, nous montrerons comment paralléliser cette technique avant de terminer par la présentation de quelques-unes de ses applications pour les problèmes de l'affectation quadratique et de l'élaboration de tournées de véhicules.

2.2. Méta-heuristiques à mémoire

Dans cette section, nous allons passer en revue quelques méta-heuristiques dotées d'une mémoire. Précisons d'emblée que nous faisons une interprétation assez large des formes que peut prendre une mémoire : par exemple, nous considérons que la population d'un algorithme génétique ou d'une recherche par dispersion est une forme de mémoire, alors que d'autres auteurs [GLO97c] envisagent les algorithmes génétiques comme des méthodes sans mémoire.

2.2.1. Algorithmes génétiques

L'idée de base des algorithmes génétiques est de simuler le processus d'évolution des espèces sexuées. Une des théories de l'évolution stipule que deux mécanismes fondamentaux entrent en jeu dans la création d'un nouvel individu lors d'une reproduction sexuée. Le premier mécanisme correspond au croisement des deux parents qui vise à combiner deux moitiés de leur patrimoine génétique respectif pour constituer le patrimoine génétique de l'enfant. Le second mécanisme consiste dans la mutation, ou modification spontanée, de quelques gènes de l'enfant. Le nouvel individu ainsi créé est différent de chacun de ses parents mais partage cependant certaines de leurs caractéristiques. Si le hasard fait que l'enfant hérite de « bonnes » caractéristiques, son espérance de vie est plus élevée et il aura par conséquent une plus grande chance de se reproduire qu'un individu taré. L'analogie entre cette théorie de l'évolution et une méta-heuristique pour l'optimisation combinatoire a été proposée par Holland [HOL75]. Elle peut être décrite comme suit :

Un individu est associé à une solution admissible du problème à résoudre. Initialement les solutions étaient codées sous la forme de vecteurs binaires. Dans les applications les plus récentes — hérésie notoire pour les puristes — les solutions sont représentées de manière naturelle, par exemple sous la forme d'une permutation pour le problème du voyageur de commerce ou de l'affectation quadratique. L'opérateur de croisement, qui consistait au départ à échanger des sous-chaînes des vecteurs correspondant aux codes des deux parents, est actuellement souvent remplacé par un opérateur mieux adapté au problème à résoudre. La mutation est un opérateur qui était invoqué occasionnellement et qui consistait à changer quelques éléments du vecteur-enfant. De nos jours, l'opérateur de mutation est parfois beaucoup plus complexe, allant jusqu'à effectuer une recherche avec tabous.

Une version élémentaire d'un algorithme génétique peut se formuler schématiquement comme suit :

Algorithme génétique

```
1) Choisir un code binaire pour représenter une solu-
   tion et générer une population de vecteurs binaires.
2) Répéter les pas suivants, tant qu'un critère d'arrêt
   n'est pas vérifié.
   2a) Sélectionner deux vecteurs dans la population à
       l'aide d'un opérateur de sélection.
   2b) Combiner les deux vecteurs-parents à l'aide
       d'un opérateur de croisement pour obtenir un
       vecteur-enfant.
   2c) Éventuellement, appliquer un opérateur de muta-
       tion au vecteur-enfant.
   2d) Évaluer la qualité de la solution-enfant.
   2e) Insérer l'enfant dans la population.
   2f) Éventuellement, éliminer des vecteurs de la
       population avec un opérateur d'élimination.
```

La formulation de cet algorithme appelle quelques remarques : tout d'abord, il est souvent peu naturel de coder une solution sous la forme d'un vecteur binaire et suivant le schéma de codage choisi, l'algorithme peut produire des résultats très différents. Une telle propriété n'est pas vraiment souhaitable. Ensuite, un critère d'arrêt doit être choisi, comme c'est d'ailleurs le cas pour toutes les méta-heuristiques. Les algorithmes génétiques possèdent un critère d'arrêt naturel : la convergence de la population. Selon les opérateurs de sélection et d'élimination, on observe que les solutions de la population ont tendance à se ressembler, sinon à devenir toutes pareilles. Si l'on se trouve dans une telle situation, il convient d'arrêter la recherche. Cependant, la solution vers laquelle l'algorithme converge est souvent trouvée longtemps avant la convergence et l'on préfère généralement fixer a priori un nombre d'itérations que l'algorithme effectuera.

L'opérateur de sélection favorise typiquement l'élection des meilleures solutions de la population, ce qui permet une convergence plus rapide de l'algorithme, parfois au détriment de la qualité des solutions trouvées. Alors que dans les versions initiales le point-clé était de choisir un bon schéma de codage, les autres opérateurs faisant partie d'un ensemble standard, les implantations les plus récentes utilisent une représentation naturelle des solutions avec des opérateurs de croisement et de mutation fortement dépendants du problème à résoudre, ces dernières constituant les éléments-clé de ces nouvelles implantations.

Finalement, l'opérateur d'élimination consiste souvent à éliminer les solutions les plus mauvaises de la population, de sorte à ramener le nombre d'individus entre des valeurs préfixées.

Les implantations d'algorithmes génétiques sont innombrables et couvrent à peu près tous les domaines de l'optimisation.

2.2.2. Recherche par dispersion

La recherche par dispersion, ou « scatter search » en anglais, a été proposée par Glover [GLO77] mais cette technique n'a pas connu un succès comparable à celui des algorithmes génétiques, ce qui peut s'expliquer peut-être par le fait que la méthode n'a pas été présentée sous des traits autant aguichants. Pourtant, la recherche par dispersion est assez similaire aux algorithmes génétiques, du moins si l'on considère les implantations les plus récentes. Il n'est donc pas exagéré de prétendre que cette technique était très moderne pour son temps, même si elle n'a pas été appréciée à sa juste valeur.

La recherche par dispersion a été proposée dans le cadre de la résolution de programmes mathématiques en nombres entiers. Tout comme les algorithmes génétiques, elle est basée sur une population de solutions (vecteurs d'entiers) qui évolue dans le temps à l'aide à la fois d'un opérateur de sélection, de la combinaison linéaire de solutions de la population vise à créer une nouvelle solution provisoire (pas forcément entière ou admissible), d'un opérateur de projection permettant de rendre la solution provisoire admissible et d'opérateurs d'élimination. Ainsi, on peut appréhender la recherche par dispersion comme un algorithme génétique spécial présentant les particularités suivantes :

— Les vecteurs binaires sont remplacés par des vecteurs d'entiers.

— L'opérateur de sélection peut élire plus de deux solutions.

— L'opérateur de croisement est remplacé par une combinaison linéaire convexe ou non convexe de vecteurs.

— L'opérateur de mutation est remplacé par un opérateur de réparation ou de projection qui ramène la solution nouvellement créée dans l'espace des solutions admissibles.

Ces particularités peuvent également être vues comme des généralisations des algorithmes génétiques, qui ont été proposées et exploitées ultérieurement par divers auteurs, notamment Mühlenbein, Georges-Schleuter & Krämer [MÜH88]. Citons en particulier :

— L'usage d'opérateurs de croisement différents de l'échange de sous-chaînes.

— L'application d'une recherche locale pour améliorer la qualité des solutions produites par l'opérateur de croisement.

— L'usage de plus de deux parents pour créer l'enfant.

— La subdivision de la population à l'aide de méthodes de regroupement en lieu et place d'un opérateur d'élimination élémentaire.

Du fait de sa redécouverte récente, les applications de la recherche par dispersion sont peu nombreuses ; on peut toutefois citer, dans l'ordre chronologique, outre la publication initiale de Glover [GLO77], une référence à cette dernière dans une application à des problèmes d'élaboration de tournées [ROC95], les travaux de Fleurent et al. [FLE96] dans le cadre de l'optimisation continue sans contraintes, ceux de Kelly, Rangaswamy & Xu [KEL96] pour la phase d'apprentissage des réseaux neuronaux ainsi que la méthode de Cung et al. [CUN97] qui permet d'obtenir des solutions de très bonne qualité pour les problèmes d'affectation quadratique.

2.2.3. Recherche avec tabous

Parmi les méta-heuristiques passées en revue dans cette section, la recherche avec tabous est la seule qui a été présentée expressément avec une mémoire, ou plus exactement un ensemble de mémoires. Le terme de recherche avec tabous a été proposé pour la première fois par Glover [GLO86]. L'idée à la base de cette technique est de modifier localement une solution de manière itérative, tout en gardant une trace de ces modifications pendant un certain laps de temps, afin d'éviter de réaliser les modifications inverses susceptibles de mener à des solutions déjà visitées. Les modifications, certaines de leurs caractéristiques ou encore des solutions entières sont mémorisées dans des listes qui en interdisent l'usage pour les itérations futures, d'où l'appellation de liste de tabous.

L'analogie avec le monde réel présidant à la recherche avec tabous est l'imitation de l'être humain cherchant la solution d'un problème d'optimisation combinatoire : tout d'abord, on construit une solution initiale, même de piètre qualité, puis on l'améliore petit à petit, par des modifications locales. Ces modifications n'améliorent pas forcément la solution à tous les coups mais on cherche à diriger la recherche dans une portion prometteuse de l'espace des solutions.

Plus tard, en 1989 et 1990, Glover a proposé un certain nombre de stratégies pour diriger la recherche et la rendre plus efficace, tout en précisant que la recherche avec tabous est ouverte à toutes les stratégies utiles pour le problème spécifique que l'on veut résoudre [GLO89, GLO90]. Il faut noter que ces nouvelles idées pour diriger la recherche n'ont été utilisées que beaucoup plus tard dans les implantations pratiques. On trouve encore à l'heure actuelle des articles décrivant des méthodes n'utilisant que les quelques ingrédients proposés dans le papier initial [GLO86] (une liste de tabous à court terme, « aspiration » d'une solution interdite si elle est la meilleure connue), ce qui ne signifie pas forcément qu'elles sont mauvaises, mais elles peuvent présenter certaines faiblesses dans la résolution de problèmes un peu particuliers.

L'usage d'une mémoire à long terme s'est répandu bien plus tard. On peut en déceler un embryon dans notre implantation pour le problème de l'affectation quadratique [TAI91] où l'on utilise inconditionnellement une modification jamais élue pendant un grand nombre d'itérations. Le pas suivant s'est constitué d'une pénalisation des modifications proportionnelle à leur fréquence d'élection comme dans [TAI93, TAI94], technique rapidement adoptée par d'autres auteurs comme [GEN94]. On peut constater là un premier exemple de mémoire adaptative, utilisée dans le but de diversifier le processus de recherche pour le forcer à explorer des portions non encore examinées de l'espace des solutions.

Par la suite, on s'est rendu compte qu'il fallait donner une importance accrue à une diversification de la recherche et actuellement on fait de plus en plus appel à une mémoire adaptative pour réaliser cette diversification. Une autre composante, négligée dans les premières implantations de recherche avec tabous mais qui prend de plus en plus d'importance, est l'intensification de la recherche dans une portion jugée prometteuse de l'espace des solutions. D'une manière extrêmement schématique, une recherche avec tabous peut être formulée comme suit :

Trame d'une recherche avec tabous

1) Générer une solution initiale $\boldsymbol{s_0}$, poser $\boldsymbol{s^*} = \boldsymbol{s_0}$, initialiser k à 0 ainsi que les mémoires.
2) Répéter, tant qu'un critère d'arrêt n'est pas satisfait :
    - 2a) Choisir $\boldsymbol{s_{k+1}}$ dans le voisinage de la solution $\boldsymbol{s_k}$, en prenant les mémoires en considération.
    - 2b) Si $\boldsymbol{s_{k+1}}$ est meilleur que $\boldsymbol{s^*}$, poser $\boldsymbol{s^*} = \boldsymbol{s_{k+1}}$ et incrémenter k.
    - 2c) Mettre à jour les mémoires.

Cette trame est exprimée en termes très généraux et chacune de ses étapes peut être d'une sophistication très variable. Par exemple, le choix de s_{k+1} à l'étape 2a peut consister simplement à examiner l'ensemble des solutions que l'on peut obtenir en appliquant une modification locale et non interdite à la solution s_k, comme dans les implantations les plus élémentaires ; mais ce choix peut également résulter d'un processus beaucoup plus complexe comme dans [ROC95] où l'on construit une nouvelle solution de toutes pièces en s'aidant des solutions précédemment produites par la recherche, solution qui est améliorée par une recherche avec tabous plus élémentaire. Ce mécanisme permet de réaliser à la fois une intensification et une diversification de la recherche.

2.2.4. Colonies de fourmis

L'idée de s'inspirer du comportement des fourmis pour trouver de bonnes solutions à des problèmes d'optimisation combinatoire a été proposée par Colorni, Dorigo &

Maniezzo [COL91]. Le principe de cette méta-heuristique est basé sur la manière dont les fourmis cherchent leur nourriture et retrouvent leur chemin pour retourner dans la fourmilière. Initialement, les fourmis explorent les environs de leur nid de manière aléatoire. Sitôt qu'une source de nourriture est repérée par une fourmi, son intérêt est évalué (quantité, qualité) et la fourmi ramène un peu de nourriture au nid. Pour retrouver son chemin, elle a pris soin de laisser derrière elle une phéromone, trace chimique qu'elle arrive à détecter. Durant le chemin du retour, elle dépose également une quantité de phéromone dépendant de l'intérêt de la source de nourriture. Comme toutes les fourmis font de même, les traces laissées augmentent plus rapidement pour les sources de nourritures proches de la fourmilière, et lorsque plusieurs traces mènent à la même source, les traces correspondant aux chemins les plus courts sont renforcées à un rythme plus élevé. Il en résulte qu'après un certain temps, les chemins les plus rapides menant aux sources de nourriture les plus importantes sont marqués par de fortes traces. De cette manière, les fourmis arrivent à optimiser leurs déplacements.

Pour transposer ce comportement à un algorithme général d'optimisation combinatoire, on fait une analogie entre l'aire dans laquelle les fourmis cherchent de la nourriture et l'ensemble des solutions admissibles du problème, entre la quantité ou la qualité de la nourriture et la fonction-objectif à optimiser et enfin entre les traces et une mémoire adaptative. Une description détaillée de ces analogies pour le problème du voyageur de commerce peut être trouvée dans [DOR96]. Très schématiquement, un algorithme basé sur une colonie de fourmis peut être décrit comme suit :

Colonie de fourmis

```
1) Initialiser les traces.
2) Répéter en parallèle pour chacune des p fourmis et
   tant qu'un critère d'arrêt n'est pas satisfait :
    2a) Construire une nouvelle solution à l'aide des
        informations contenues dans les traces et une
```

```
    fonction d'évaluation partielle.
2b) Évaluer la qualité de la solution.
2c) Mettre à jour les traces.
```

Une particularité de cet algorithme est qu'il a été pensé pour une exécution concurrente, contrairement à la recherche avec tabous ou le recuit simulé. Le point crucial de cette méta-heuristique se compose de la définition des traces et de leur mise à jour selon le problème à résoudre. Pour un voyageur de commerce par exemple, on peut prévoir une trace pour chaque paire de villes. Au départ, deux mécanismes d'utilisation des traces ont été imaginés : l'exploration et l'exploitation. L'exploration consiste à choisir la prochaine composante utilisée pour construire la solution (par exemple une arête dans le cas du voyageur de commerce) avec une probabilité proportionnelle à la valeur de la trace associée à cette composante. L'exploitation consiste à choisir la composante optimisant une fonction qui marie la valeur de la trace et la fonction d'évaluation partielle (par exemple, pour le voyageur de commerce, une combinaison linéaire de la valeur de la trace et de la longueur de l'arête correspondante).

La mise à jour des traces a été proposée comme suit : tout d'abord, la valeur de toutes les traces est diminuée, pour simuler le phénomène d'évaporation des phéromones. Ensuite, les traces correspondant aux composantes choisies dans la construction de la solution sont renforcées d'une valeur croissant avec la qualité de la solution. Naturellement, les algorithmes basés sur une colonie de fourmis ont aussi évolué. Pour les faire converger plus rapidement, on peut par exemple ne faire la mise à jour des traces que pour la meilleure solution produite par les fourmis à l'itération courante ou même durant toute la recherche. Toutes les implantations récentes [DOR97, GAM97, STÜ97, TAI97b] améliorent la solution produite par chaque fourmi à l'aide d'une recherche locale gloutonne.

2.3. Programmation à mémoire adaptative

Bien que les méta-heuristiques présentées ci-dessus soient décrites de manière très différentes et qu'elles semblent basées sur des principes distincts, elles sont en réalité très semblables, du moins si l'on considère les implantations les plus récentes et les plus efficaces. En effet, ces implantations fonctionnent toutes selon les mêmes principes. En d'autres termes, on observe une unification des diverses méta-heuristiques au point qu'il devient difficile de les distinguer les unes des autres. Par exemple, la population d'un algorithme génétique ou d'une recherche par dispersion ou encore les traces d'une colonie de fourmis peuvent être vues comme une mémoire spéciale d'une recherche avec tabous. Inversement, une recherche avec tabous, qui construit périodiquement une nouvelle solution en utilisant une mémoire, peut être assimilée à une colonie de fourmis ou à une recherche par dispersion. On observe donc que les meilleures méthodes générales de résolution de problèmes d'optimisation combinatoire présentent les mêmes caractéristiques :

1) Les solutions générées par la recherche (ou bien une structure de données spéciale agrégeant des particularités de ces solutions) sont mémorisées.
2) Une solution provisoire est construite en consultant la mémoire.
3) La solution provisoire est améliorée à l'aide d'une recherche locale gloutonne ou avec une autre méta-heuristique de base.
4) La nouvelle solution est utilisée pour mettre à jour la mémoire, pour adapter cette dernière aux nouvelles connaissances que la solution apporte.

Ainsi, il est très naturel de parler de programmation à mémoire adaptative pour qualifier ce type de méta-heuristique. Il est certainement plus logique et moins restrictif de parler de mémoire que de tabous pour une recherche avec tabous car la part des interdits est souvent petite par rapport aux autres composantes, en ce qui concerne des implantations relativement avancées. De même, il est aussi légitime de considérer les traces d'une colonie de fourmis ou les solutions de la population d'un algorithme génétique ou d'une recherche par dispersion comme une forme de mémoire qui s'adapte aux nouvelles connaissances acquises en cours de recherche.

Seule la recherche avec tabous, parmi les méthodes à mémoire considérées dans cette section, n'a pas explicitement été conçue initialement avec une construction de toutes pièces d'une nouvelle solution, alors que toutes les autres incluent cela comme principe de base. Cependant, bien des recherches avec tabous incluent un redémarrage du processus à partir d'une nouvelle solution, que ce soit dans un but de diversification, par exemple par la réalisation d'un certain nombre de modifications aléatoires comme dans la recherche avec tabous réactive de Battiti & Tecchiolli [BAT94], ou dans un but d'intensification, par exemple en repartant périodiquement des voisins des meilleures solutions trouvées par la recherche [NOW96].

Concernant l'incorporation d'une procédure de recherche locale dans le processus, il est clair que la recherche avec tabous est elle-même une recherche locale. La recherche par dispersion comprend dans sa conception une procédure spéciale de réparation de la solution provisoire, afin de ramener cette dernière dans l'ensemble des solutions admissibles. Dans sa mouture la plus récente [GLO97a], elle incorpore expressément une procédure d'amélioration ; elle est donc parfaitement calquée sur le modèle de programmation par mémoire adaptative proposé ici.

On pourrait objecter que les algorithmes génétiques et les méthodes basées sur les colonies de fourmis n'ont pas été élaborés initialement avec une procédure de recherche locale. On observe toutefois un usage de plus en plus systématique d'une telle procédure dans les implantations récentes, car on s'est rendu compte que cela permettait d'améliorer notablement la qualité des solutions trouvées. En particulier, les algorithmes génétiques sont actuellement presque systématiquement hybridés avec une autre méta-heuristique, telle qu'un algorithme glouton d'optimisation, un recuit simulé ou une recherche avec tabous. De même, les dernières évolutions des colonies de fourmis sont aussi dotées d'une recherche locale. Ainsi, les trames des méthodes examinées dans la section précédente sont similaires et peuvent se formuler comme suit :

Programmation à mémoire adaptative

```
1) Initialiser la mémoire.
2) Répéter, tant qu'un critère d'arrêt n'est pas
   satisfait :
   2a) Générer une solution provisoire μ à partir des
       informations contenues dans la mémoire.
   2b) Améliorer μ à l'aide d'une recherche locale pour
       obtenir une solution π.
   2c) Mettre à jour la mémoire en incorporant les élé-
       ments d'information que contient π.
```

2.4. Application de programmes à mémoire adaptative

Nous donnerons dans cette section des applications de la programmation à mémoire adaptative sur deux types de problèmes : l'affectation quadratique et l'élaboration de tournées de véhicules. Bien que la trame de l'algorithme général que nous avons donnée pour la programmation à mémoire adaptative soit très simple, on peut en réaliser des implantations très diverses, suivant les choix opérés à chacune des étapes de l'algorithme.

2.4.1. Problème de l'affectation quadratique

Le problème de l'affectation quadratique (QAP) peut être formulé comme suit : étant donnés n unités et n sites où elles peuvent être placées, connaissant le flot f_{ij} circulant entre les unités i et j (i, $j = 1, \ldots, n$) et la distance d_{rs} entre les sites r et s (r, $s = 1, \ldots, n$), on cherche un placement des unités sur les sites minimisant la somme totale des produits flots $\times$ distances. Mathématiquement, on cherche une permutation π^* de $\Pi(n)$, l'ensemble des permutations de n éléments, minimisant :

$$\min_{\pi \in \Pi(n)} f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi_i \pi_j}$$

Ce problème a été posé pour la première fois par Koopmans & Beckman [KOO57] ; il est NP-difficile, même si l'on ne désire trouver qu'une solution approchée à ε près [SHA76]. Beaucoup de problèmes peuvent être modélisés comme un QAP, à commencer par le problème du voyageur de commerce, la conception de bâtiments [ELS77], l'équilibrage de turbines [LAP88], la génération de trames de gris [TAI95], etc. Il s'agit sans doute d'un des problèmes d'optimisation combinatoire les plus difficiles puisque les méthodes exactes actuelles résolvent avec peine des problèmes de taille $n = 20$. Cette difficulté n'est pas étrangère à notre ignorance de bonnes bornes inférieures.

Un grand nombre de méthodes ont donc été mises au point pour obtenir des solutions approchées au QAP. Parmi les plus efficaces, nous pouvons citer :

— La recherche avec tabous réactive (RTS) de Battiti & Tecchiolli [BAT94],

— notre recherche avec tabous (TS) [TAI91],

— la recherche par dispersion (SS) de Cung & al. [CUN97],

— les algorithmes génétiques hybrides de Fleurent & Ferland (GTSH) [FLE94] et de Taillard & Gambardella (GDH) [TAI97b],

— plusieurs méthodes basées sur les colonies de fourmis dues à Stützle & Hoos (MMAS) [STÜ97], Gambardella, Taillard & Dorigo (HAS-QAP) [GAM97], Taillard & Gambardella (FANT) [TAI97b].

Toutes ces méthodes utilisent une mémoire, sous une forme ou une autre. Les deux recherches avec tabous TS et RTS utilisent cette mémoire pour empêcher la visite de certaines solutions, justifiant pleinement l'appellation de recherche avec tabous. Ces deux méthodes sont particulièrement efficaces pour les problèmes générés aléatoirement, uniformément. Par contre leurs performances sur des problèmes irréguliers sont médiocres [TAI95, GAM97, TAI97b]. Les autres méthodes citées ci-dessus utilisent une mémoire pour construire une solution provisoire, solution qui sera ensuite améliorée par une recherche locale. Il s'agit donc de procédures dont le fonctionnement est calqué sur la trame de la programmation à mémoire adaptative de la section 2.3. Nous allons donc présenter plus en détail les choix qui ont été faits pour l'implantation de ces méthodes, c'est à dire en spécifiant : 1) le type de mémoire utilisée avec son processus d'adaptation, 2) la procédure de construction d'une solution provisoire et 3) la procédure de recherche locale.

1) Mémoires

Une solution d'un QAP pouvant être représentée par une permutation π de n éléments, où π_i indique l'emplacement de l'unité i, la plupart des méthodes à mémoire adaptative utilisent une matrice de taille $n \times n$ pour implanter la mémoire, l'entrée (i, j) de cette matrice étant une statistique mesurant l'intérêt de poser $\pi_i = j$. Deux méthodes, (GTSH et GDH) utilisent une population formée des meilleures solutions trouvées par la recherche, dans l'esprit des algorithmes génétiques. Finalement, deux méthodes (SS, HAS-QAP) utilisent à la fois une (petite) population de solutions et une matrice statistique sur les emplacements des unités. À chaque itération d'une procédure à mémoire adaptative, une solution π est produite (éventuellement p solutions $\pi^1, \ldots \pi^p$ sont produites simultanément). On considère de plus π^*, la meilleure solution trouvée par la procédure jusqu'à l'itération courante. La mise à jour de la mémoire se fait de la manière suivante :

SS, GTSH et GDH

π est insérée dans la population et la plus mauvaise solution de la population en est retirée. SS maintient de plus une matrice qui enregistre le nombre de fois que chaque unité a occupé un site donné.

FANT, HAS-QAP, MMAS

Ces trois méthodes ont été élaborées dans le cadre des colonies de fourmis et ont donc pour mémoire une matrice ***T*** de traces dont l'entrée τ_{ij} indique l'intérêt de poser $\pi_\mathrm{i} = j$. Cet intérêt se mesure par une statistique sur les solutions trouvées par la recherche, statistique qui varie selon la méthode.

Par exemple, HAS-QAP initialise toutes les entrées à la même valeur $\tau_\mathrm{ij} = \tau_0$, $(1 \le i, j \le n)$, où t_0 est un paramètre de la méthode. Après chaque itération, toutes les entrées de la matrice sont affaiblies d'un facteur α $(0 < \alpha < 1)$ en posant $\tau_\mathrm{ij} = \alpha\tau_{ij}$, où α est un second paramètre représentant l'évaporation des phéromones. Finalement, les entrées $\tau_{i\pi_i^*}$ $(1 \le i \le n)$, associées à la meilleure des solutions trouvées par la recherche, π^*, sont augmentées d'une valeur $\tau_0(1-\alpha)/f(\pi^*)$. De plus, deux mécanismes utilisés exceptionnellement sont prévus : le premier pour éviter la stagnation de la recherche à cause de la convergence de la mémoire adaptative, le second pour accélérer cette convergence lorsque cela se justifie. Cette méthode utilise également une petite population de solutions comme mémoire adaptative. À chaque itération, certaines solutions de cette population sont changées. Par exemple, lorsqu'on suspecte une convergence de l'algorithme, toutes les solutions sauf la meilleure sont éliminées et remplacées par des solutions aléatoires. Inversement, lorsque l'on vient d'améliorer la meilleure des solutions (ce qui indique que le processus n'a pas convergé), on n'élimine de la population que les solutions qui sont plus mauvaises que celles nouvellement générées à l'itération courante.

L'idée à la base de MMAS est de limiter les valeurs possibles des entrées de la matrice ***T*** entre deux valeurs, τ_{min} et τ_{max}, afin de prévenir une convergence prématurée de la recherche. Initialement, toutes les entrées de la matrice sont mises à τ_{max} ; elles sont affaiblies d'un facteur α à chaque itération et les entrées $\tau_{i\pi_i}$ sont renforcées d'une quantité $Q/\tau_{i\pi_i}$, où Q est un paramètre et π la meilleure solution produite à l'itération courante.

FANT initialise toutes les entrées de la matrice avec une valeur *r*. À chaque itération, les entrées $\tau_{i\pi_i}$ sont incrémentées de *r* et les entrées $\tau_{i\pi_i^*}$ sont incrémentées de *R*, où *r* et *R* sont deux paramètres. Nous avons également prévu un mécanisme pour éviter la convergence prématurée de la méthode : lorsque les entrées $\tau_{i\pi_i^*}$ sont si grandes que l'information contenue dans $\boldsymbol{T}$ ne diffère pas significativement de π^*, la valeur de *r* est augmentée et $\boldsymbol{T}$ est réinitialisé ; lorsque π^* est améliorée, *r* reprend sa valeur initiale.

2) Solution provisoire

La procédure de construction d'une solution à partir de la mémoire est fortement dépendante du type d'informations qui y sont stockées. Les méthodes travaillant avec une population de solutions (GDH, GTSH et HAS-QAP) auront donc une procédure de génération très différente de celles construisant une solution à l'aide d'une matrice-statistique (SS, MMAS, FANT).

GDH et GTSH construisent une nouvelle solution en sélectionnant deux solutions de la population et en appliquant un opérateur de croisement spécialisé pour les permutations dû à Tate & Smith [TAT95].

FANT et MMAS construisent une solution provisoire μ en choisissant séquentiellement les entrées de μ aléatoirement avec une probabilité dépendant des entrées de la matrice $\boldsymbol{T}$. MMAS répète cette construction *p* fois (où *p* est un paramètre de la méthode, représentant le nombre de fourmis) ; ces *p* solutions provisoires sont évaluées et seule la meilleure est retenue pour amélioration par la recherche locale.

HAS-QAP maintient une population de *p* solutions. Cette méthode se distingue des autres car elle ne construit pas une solution de toutes pièces à partir de la mémoire, mais elle modifie toutes les solutions de la population en faisant subir *s* modifications à chacune de ces solutions, où *s* est un paramètre de la méthode. Chacune de ces modifications consiste à échanger deux unités de site, le premier étant choisi aléatoirement, uniformément, et le second avec une probabilité qui dépend des entrées de la matrice-mémoire.

3) Procédure d'amélioration

FANT, HAS-QAP et GDH utilisent toutes la même procédure d'amélioration qui évalue chaque paire d'échange de deux unités et effectue immédiatement les échanges qui améliorent la solution. Cette procédure est répétée si une amélioration a été trouvée ; elle est très rapide, quoiqu'en $O(n^3)$.

GTSH utilise notre méthode TS comme procédure d'amélioration, chaque solution provisoire étant améliorée par $4n$ itérations de TS. La complexité de cette procédure est également $O(n^3)$, mais elle est approximativement 8 fois plus lente que la procédure utilisée dans FANT, HAS-QAP et GHD.

Il existe deux versions de MMAS, la première utilisant $4n$ itérations de TS comme procédure d'amélioration et la seconde effectuant une descente vers le premier optimum local relativement à l'échange de deux unités. SS utilise une recherche avec tabous comme procédure d'amélioration.

2.4.2. Efficacité des méthodes à mémoire adaptative pour le QAP

Dans [GAM97] on trouvera une comparaison de HAS-QAP avec d'autres méthodes : TS, RTS, un recuit simulé, GTSH. Dans [TAI97b], on confronte FANT et GDH à HAS-QAP, TS et GTSH. De plus, [STÜ97] compare deux versions de MMAS à HAS-QAP et GTSH. Finalement, nous comparons, entre autres, TS, RTS et GTSH dans [TAI95]. Il est donc possible d'avoir une bonne idée des performances relatives de toutes ces méthodes et tous les auteurs s'accordent sur un certain nombre de points :

— Lorsque les données présentent une structure marquée ou sont très irrégulières, par exemple lorsque les unités forment des groupes ou lorsque la matrice de flots a des coefficients très variables, avec un grand nombre d'entrées nulles, la programmation à mémoire adaptative réussit à identifier la structure des bonnes solutions alors que les recherches avec tabous, ou le recuit simulé quant à eux, ont beaucoup plus de peine à s'en rapprocher, parce qu'ils restent souvent piégés dans des optima locaux de piètre qualité.

— Inversement, lorsque les problèmes sont peu structurés, par exemple ceux générés aléatoirement, uniformément, les recherches avec tabous se placent parmi les méthodes les plus performantes alors que les autres programmes à mémoire adaptative ont de la peine à identifier la structure d'une solution optimale, ce qui n'est pas étonnant si l'on sait que les optima locaux de tels problèmes sont répartis assez uniformément dans l'espace des solutions, comme nous avons pu l'établir en étudiant leur entropie [TAI95].

Notons que nous avons également pu observer des comportements similaires en comparant recherche avec tabous et programmes à mémoire adaptative pour des problèmes d'élaboration de tournées de véhicules [ROC95].

Les comparaisons directes des diverses méthodes à mémoire adaptative FANT, HAS-QAP, GDH et MMAS (variante « rapide » avec méthode de descente) révèlent des performances très voisines, bien que leurs implantations soient assez différentes. Il semble donc que la conception d'une méthode à mémoire adaptative soit relativement aisée à réaliser pour le QAP et que le type de mémoire utilisé, et la procédure de construction d'une solution provisoire, ne soient pas très sensibles aux options choisies, pour autant que l'on prenne soin de prévoir, d'une part, un mécanisme de diversification au cas où la méthode convergerait prématurément ; d'autre part, il faut également s'assurer que les informations contenues dans la mémoire convergent et permettent de construire des solutions comportant une bonne structure générale.

Pour les problèmes irréguliers, il convient de mentionner que les programmes à mémoire adaptative dont nous venons de discuter sont les meilleures méthodes à l'heure actuelle pour le problème d'affectation quadratique [GAM97, TAI97b, STÜ97, CUN97].

2.4.3. Problème d'élaboration de tournées

Il existe un grand nombre de variantes de problèmes d'élaboration de tournées de véhicules. Dans une de ses formes les plus simples, ce problème consiste en un ensemble de n clients qui demandent un bien en quantité q_i ($i = 1, \ldots, n$). Pour satisfaire ces demandes, on dispose d'un véhicule de capacité Q qui doit se réapprovisionner à partir d'un dépôt unique. Connaissant les distances entre chaque paire de clients et entre le dépôt et les clients, on cherche des tournées de longueur totale minimale telles que la quantité à livrer dans chaque tournée soit au plus égale à Q. À partir de ce modèle de base, que l'on nommera A, il est possible d'ajouter des contraintes ou des ressources pour arriver à des modèles plus compliqués mais plus proches de la réalité :

- B la durée de chaque tournée est limitée, et on a un temps de service pour chaque client.
- C les livraisons s'organisent en journées de travail d'une durée limitée L et toutes les commandes doivent avoir été livrées dans un laps de temps de K jours ; il est donc possible de faire plusieurs tournées par jour ; afin d'assurer l'existence d'une solution admissible, il est permis de dépasser la limite de temps L moyennant une pénalité proportionnelle aux heures supplémentaires.
- D on dispose d'une flotte hétérogène de véhicules ; chacun étant spécifié par sa capacité, son coût fixe d'engagement et un coût variable d'utilisation dépendant de la longueur du trajet effectué.
- E le nombre de tournées est limité et on désire minimiser la longueur de la plus longue tournée.
- F les livraisons s'organisent en K journées de travail et on désire minimiser la durée de la plus longue journée de travail.
- G les clients spécifient une fenêtre de temps durant laquelle ils peuvent recevoir leur livraison. On désire engager un nombre minimum de véhicules pour réaliser l'ensemble des livraisons.

Pour tous ces problèmes, nous avons mis au point des procédures à mémoire adaptative qui utilisent toutes la même forme de mémoire et le même mode de construction des solutions provisoires, qualifié aussi de construction de vocabulaire dans [GLO97c].

La mémoire est constituée de l'ensemble de toutes les tournées individuelles contenues dans les solutions trouvées par la recherche. La procédure de construction d'une solution provisoire consiste à choisir une tournée aléatoirement dans la mémoire telle qu'aucun de ses clients ne soient compris dans les tournées précédemment sélectionnées. La probabilité de choisir une tournée donnée dépend de la qualité de la solution dans laquelle on retrouve cette tournée. Ainsi, la mémoire peut contenir plusieurs fois la même tournée avec plusieurs évaluations différentes. La procédure de construction d'une solution provisoire ne produit pas une solution admissible à chaque appel : il est en effet très probable que des clients ne soient pas touchés par les tournées choisies mais que la mémoire ne contienne pas de tournée constituée uniquement d'un sous-ensemble des clients non touchés. Il est par conséquent très important de concevoir une procédure de recherche locale qui puisse rendre admissibles de telles solutions. Pour les variantes de problèmes A à F, nous avons utilisé une recherche avec tabous de base décrite dans [TAI93].

Pour la variante G, nous avons recouru à d'autres procédures d'optimisation, basées elles aussi sur une recherche avec tabous, mais capables de prendre en considération les fenêtres de temps. La première de ces procédures est la simplification d'une méthode mise au point pour une application industrielle qui tient compte d'autres contraintes comme des fenêtres de temps multiples, les pauses des chauffeurs et l'inaccessibilité de certains clients par certains véhicules [ROC94]. La deuxième procédure prend en considération des fenêtres de temps souples, c'est-à-dire qu'il est possible d'arriver en retard chez un client moyennant une pénalité. Cette procédure se fonde sur une recherche avec tabous utilisant un voisinage original : l'échange de sous-tournées. Ce voisinage en contient plusieurs autres communément utilisés pour les problèmes d'élaboration de tournées, comme l'insertion d'un client dans une tournée, l'échange de deux clients appartenant à des tournées différentes, la concaténation de deux tournées, etc.

L'algorithme général de résolution de ces problèmes de distribution peut être esquissé comme suit :

Programme à mémoire adaptative pour problème d'élaboration de tournées

1) $M \leftarrow \varnothing$.
2) Répéter, tant qu'un critère d'arrêt n'est pas satisfait :
   - 2a) $M' \leftarrow M$, $s \leftarrow \varnothing$.
   - 2b) Répéter, tant que $M' \neq \varnothing$:
     - 2b1) Choisir une tournée $T \in M'$, aléatoirement.
     - 2b2) Poser $s' \leftarrow s' \cup \{T\}$.
     - 2b3) Pour toute tournée $T' \in M'$ telle que $T \cap T' \neq \varnothing$, poser $M' \leftarrow M' \setminus \{T'\}$.
   - 2c) Compléter la solution partielle s' et l'améliorer à l'aide d'une recherche locale pour obtenir une solution s.
   - 2d) Pour toute tournée T de s, poser : $M \leftarrow M \cup \{T\}$.
3) Trouver une solution s^* aussi bonne que possible en considérant les tournées contenues dans M.

Ce programme appelle trois remarques : premièrement mentionnons qu'il peut s'adapter pour la résolution d'autres problèmes dont les solutions sont constituées d'une partition d'un ensemble d'élements, comme par exemple la coloration des sommets d'un graphe.

Deuxièmement, dans le cas de problèmes avec flotte hétérogène, on doit considérer une mémoire différente pour chaque type de véhicules. En effet, nous n'avons implanté que des recherches locales pour des problèmes homogènes, donc nous répétons l'étape 2 pour chaque type de véhicule, comme si l'on avait une flotte homogène.

Troisièmement, l'étape 3 consiste à résoudre un problème d'optimisation NP-difficile : pour les problèmes de type A, B, D et G, il s'agit de problèmes de partition d'ensemble ; pour les problèmes de type C, E et F, le problème de partition d'ensemble est couplé à un problème de mise en boîte. Il existe actuellement des méthodes exactes assez efficaces pour résoudre le problème de partition d'ensemble ; même un logiciel général de programmation linéaire en nombres entiers permet d'en résoudre des exemples de taille acceptable (quelques centaines de clients et plusieurs centaines à quelques milliers de tournées contenues dans la mémoire). Pour les types de problèmes requérant une mise en boîte, nous avons recouru à la méthode suivante [TAI96c, GOL97] : on commence par résoudre le problème de partition en énumérant l'ensemble de toutes les solutions d'un problème de type A que l'on peut construire à l'aide des tournées contenues dans la mémoire ; ensuite, dans le cas de la variante E, on met une tournée par boîte pour chaque solution énumérée et on retient la meilleure solution admissible ainsi trouvée ; dans le cas des variantes C et F, on cherche pour chacune de ces solutions une mise en boîte des tournées (i.e. un regroupement par journée de travail) à l'aide d'une heuristique simple (insertion gloutonne suivie éventuellement d'une descente dans le premier optimum local relativement à l'échange de jours de deux tournées).

À l'aide de ce type de programme à mémoire adaptative, nous avons pu mettre au point des méthodes très efficaces pour tous les types de problèmes d'élaboration de tournées mentionnés ci-dessus. Tout d'abord, nous montrons dans [ROC95] que l'encapsulation d'une recherche avec tabous à l'intérieur de ce schéma de programme à mémoire adaptative permet d'obtenir une méthode sensiblement plus robuste que la recherche avec tabous initiale. Les meilleures solutions connues d'un nombre conséquent d'exemples de problèmes de type A, B et F de la littérature ont ainsi pu être améliorées ou obtenues.

Dans [TAI96a], nous proposons une méthode pour les problèmes avec flotte hétérogène. Nous avons réussi à obtenir ou à améliorer toutes les meilleures solutions connues de problèmes de la littérature à l'aide de cette technique. Dans [GOL97], nous décrivons plus en détail les méthodes conçues pour les variantes de type E et F. Dans [TAI96c], nous montrons que les solutions trouvées par la programmation à mémoire adaptative sont souvent très proches d'une borne inférieure, ou éventuellement que les dépassements de la durée limite L des problèmes de type C sont faibles.

Dans [TAI97a], nous évaluons les performances d'un nouveau type de voisinage pour les problèmes de distribution avec application à une variante du type G où les fenêtres de temps sont souples. Ce nouveau voisinage est utilisé dans une recherche avec tabous servant de procédure d'amélioration d'un programme à mémoire adaptative. Nous avons ensuite parallélisé cette méthode dans [BAD97], (voir chapitre 4 pour les techniques de parallélisation) avant de l'adapter pour la résolution de problèmes dynamiques dans [GEN96a, GEN96b].

Un des avantages de la programmation à mémoire adaptative dont nous n'avons pas encore parlé est son adéquation au traitement de problèmes dynamiques. En effet, les modifications des données du problème peuvent être intégrées rapidement et les informations contenues dans la mémoire restent pertinentes si le problème n'a pas été trop modifié et que la structure des bonnes solutions ne change pas significativement. Bien que la recherche locale soit la partie la plus gourmande en temps de calcul, un appel ne prend pas beaucoup de temps en valeur absolue. Par conséquent, il est possible d'intégrer en temps réel les nouvelles données du problème, par exemple un nouveau client pour l'élaboration de tournées de véhicules. Dans [GEN96a, GEN96b], nous avons montré que la programmation à mémoire adaptative permettait d'obtenir de meilleures solutions que des heuristiques simples comme la meilleure insertion possible dans la solution courante ou la reconstruction complète d'une solution avec le nouveau client, suivies d'une procédure de recherche locale : la qualité des tournées obtenues par programmation adaptative, que ce soit en utilisant une méthode de descente ou une recherche avec tabous comme procédure de recherche locale, est nettement meilleure, tant du point de vue de la longueur des tournées, du nombre de clients qui ont pu être insérés avec succès, que des éventuels retards dans les visites. De plus, le fait que la programmation à mémoire adaptative soit facilement parallélisable renforce encore ses performances sur les problèmes dynamiques, ainsi que nous l'avons également montré dans les articles cités.

3. MÉTHODES PSEUDO-GLOUTONNES.

Suivant le voisinage que l'on choisit, une simple méthode de descente peut produire d'excellentes solutions ; ceci est connu depuis longtemps et on peut citer en exemple le problème du voyageur de commerce associé au voisinage de Lin & Kernighan [LIN73]. Pour d'autres problèmes en revanche, une méthode de descente simple associée à un voisinage V_1 de taille limitée fonctionne très mal. Pour s'affranchir des optima locaux associés à V_1, il faut pouvoir effectuer des « sauts » dans l'espace des solutions, en utilisant un voisinage étendu V_2, qu'il ne sera plus forcément possible d'examiner intégralement et dont les éléments ne pourront être obtenus à partir de V_1 qu'en effectuant un nombre relativement élevé de transitions simples, certaines d'entre elles dégradant la qualité de la solution, d'où notre appellation de « pseudo-glouton ».

On peut implanter un algorithme pseudo-glouton de diverses manières que nous allons illustrer premièrement sur des problèmes d'élaboration de tournées de véhicules et secondement en classification avec centroïdes. Dans le premier cas, il n'est pas possible d'énumérer complètement (ou même très partiellement) en un temps raisonnable le voisinage étendu V_2. On recourt donc à une recherche avec tabous basée sur un voisinage simple V_1 pour trouver une solution voisine dans V_2 de qualité supérieure. Dans le second cas, à partir d'un optimum local relativement à V_1, on effectue un saut dans V_2 avant de rechercher un nouvel optimum local relativement à V_1; ce dernier est accepté seulement s'il est meilleur que l'optimum local de départ.

Remarquons que les méthodes pseudo-gloutonnes peuvent également englober des implantations particulières des oscillations stratégiques ou des listes de candidats de Glover [GLO89, 90], la recherche à voisinage variable de Mladenovic [MLA95] voire dans certains cas des méthodes de décomposition.

3.1. Algorithme pseudo-glouton pour problèmes d'élaboration de tournées

Nous avons pour la première fois implanté un algorithme pseudo-glouton pour des problèmes de distribution de type A et B (voir chapitre précédent) dans [TAI93]. Nous avons ensuite utilisé un schéma semblable pour les autres types de problèmes présentés dans la section 2.4.3.

La base de notre algorithme pseudo-glouton repose sur la constatation suivante : la résolution de problèmes d'élaboration de tournées ne comportant que peu de véhicules est souvent aisée avec une recherche avec tabous élémentaire. Connaissant une solution d'un problème comportant un nombre élevé de tournées, on peut essayer de l'améliorer en ne considérant qu'un sous-ensemble de K de ses tournées que l'on optimise. Si l'on trouve une amélioration, on l'effectue, sinon, on retourne à la meilleure solution trouvée jusque là et l'on considère un autre sous-ensemble de K tournées.

Nous avons donc ainsi un voisinage étendu $V_2(K)$ qui comporte toutes les solutions que l'on peut obtenir en modifiant au plus K tournées. Il est clair qu'avec un tel voisinage, un optimum local sera généralement meilleur que les optima locaux associés à des voisinages beaucoup plus restreints couramment utilisés dans les méta-heuristiques, consistant à ne modifier que légèrement quelques trajets.

Cependant, $V_2(K)$ comporte un grand nombre de solutions voisines peu intéressantes a priori, par exemple celles qui peuvent être construites en modifiant $k' \leq K$ tournées qui, prises isolément, sont déjà optimales. De plus, même s'il était possible de trouver très rapidement l'optimum d'un problème à K tournées, il faudrait tout de même examiner C_K^m sous-problèmes pour s'assurer que l'on se trouve bien dans un optimum local par rapport au voisinage $V_2(K)$; nous nommons ici m le nombre de tournées d'une solution du problème originel.

Considérant que les clients des problèmes réels de distribution sont disséminés sur un plan (e. g. une portion de la surface terrestre), nous recourons à la technique suivante pour réduire le nombre de sous-problèmes que l'on essaie d'améliorer.

Tout d'abord, nous calculons le centre de gravité de chacune des tournées d'une solution, en prenant en considération la position et le poids des clients. Les tournées sont triées suivant les coordonnées polaires de leur centre de gravité et nous ne considérons que des sous-ensembles de K tournées dont les angles des centres sont adjacents. On évite de cette manière l'examen d'une multitude de sous-problèmes déjà résolus optimalement. Notons que si m $2K$, il est possible d'optimiser simultanément deux ou plusieurs sous-ensembles de tournées disjoints, et l'on obtient ainsi un algorithme parallèle basé sur une décomposition du problème.

Nous avons pour la première fois implanté avec succès une telle méthode pseudo-gloutonne pour des problèmes de type A et B dans [TAI93]. Dans cette implantation, pour résoudre sub-optimalement et rapidement les sous-problèmes engendrés par le voisinage $V_2(K)$, nous avons fait usage d'une recherche avec tabous relativement simple, décrite en détail dans cette dernière référence. Comme voisinage V_1, nous avons opté l'échange de clients appartenant à des tournées distinctes et le déplacement d'un client sur une autre tournée, mouvements très communément utilisés.

Sur des problèmes uniformes (coordonnées et poids des clients uniformément distribués), cette méthode permet d'obtenir des solutions d'extrêmement bonne qualité. Par contre, si les problèmes sont irréguliers, les résultats sont beaucoup moins constants et dépendent fortement de la solution initiale choisie. L'encapsulation de la procédure d'optimisation pseudo-gloutonne dans un schéma de programmation à mémoire adaptative est donc conseillé.

Pour des problèmes de type G avec fenêtres de temps souples, nous avons proposé dans [TAI97a] un autre voisinage V_1 basé sur l'échange de sous-séquences de clients appartenant à des tournées distinctes. Ce voisinage en contient plusieurs autres, notamment ceux utilisés dans [TAI93, ROC94, ROC95] lorsqu'une des séquences contient un seul client et l'autre n'en contient pas plus. Les fenêtres de temps donnant une forte structure au problème, il est également souvent profitable d'encapsuler la recherche pseudo-gloutonne dans un schéma de programmation à mémoire adaptative.

Notons finalement que lorsque les méthodes exactes auront suffisamment progressé pour pouvoir résoudre rapidement et de manière robuste des sous-problèmes jusqu'à K tournées, elles pourront se substituer à la recherche avec tabous, mais pour l'instant, cette dernière reste beaucoup plus rapide et fiable.

3.2. Application à des problèmes de classification avec centroïdes

Les problèmes de classification consistent à former des sous-ensembles d'entités, auxquelles sont associées des mesures ou des observations les décrivant, tels que les entités composant un sous-ensemble soient aussi homogènes que possible alors que les entités appartenant à des groupes distincts soient bien séparées. Ce type de problème, dont on peut déjà trouver des traces chez les Grecs, a été fort étudié au 18e siècle, en particulier par des naturalistes comme Buffon ou Linné dans le cadre de la classification des espèces animales et végétales.

Les problèmes de classification avec centroïdes mesurent l'homogénéité des groupes par rapport à des centres : chaque entité étant rattachée à un centre, on cherche à minimiser une fonction de distance entre les entités et les centres. Un problème d'optimisation combinatoire bien connu en classification avec centroïdes est celui de la p-médiane. Les n entités sont caractérisées par une matrice de distance $\boldsymbol{D} = (d_{ij})$ où

d_{ij} est la distance entre les entités i et j. Les p centres doivent être choisis parmi les n entités. Mathématiquement, ce problème se formule comme suit :

$$\begin{array}{lll} \min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} & \\ \text{s. c. :} & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & \forall i \\ & y_j - x_{ij} \geq 0 & \forall i, j, i \neq j \\ & \displaystyle\sum_{j=1}^{n} y_j \leq p & \\ & x_{ij} \in \{0, 1\} & \forall i, j \end{array}$$

Dans cette formulation, x_{ij} est une variable indiquant si l'entité i est allouée au centre placé sur l'entité j ; y_j indique si un centre doit être placé sur l'entité j. Notons que le problème de la p-médiane est NP-difficile [KAR79].

D'autres problèmes de classification, où les n entités sont caractérisées par leur position géographique $\boldsymbol{c_i}$ ainsi que par leur poids p_i ($i = 1, \ldots, n$) peuvent avoir des centres placés dans un espace E muni d'une fonction de distance $d(\cdot, \cdot)$. On cherche alors la position $\boldsymbol{x_j}$ des centres ($j = 1, \ldots, p$) minimisant la somme de la distance pondérée des entités au centre le plus proche :

$$\min_{x_j \in E, \quad j = 1, \ldots, p} \quad \sum_{i=1}^{n} \min_j \; p_i d(x_j, c_i)$$

Pour le problème de Weber, les entités sont des points du plan euclidien et la fonction de distance est la distance euclidienne habituelle. La minimisation de la somme du carré des distances est un autre problème classique. Ces trois problèmes de classification ont été approchés par des méta-heuristiques plus ou moins perfectionnées comme des recherches avec tabous [MLA96, VOß96, ROL97] ou des recherches à voisinage variable [HAN97]. Nous avons appliqué à ces trois problèmes un algorithme pseudo-glouton qui s'est révélé extrêmement compétitif relativement à la qualité des solutions.

La procédure de recherche locale élémentaire que nous avons utilisée consiste à trouver, à partir d'une solution initiale donnée, une solution stable quant à l'allocation des entités aux centres et à la localisation des centres relativement aux entités qui leur sont allouées. Il s'agit d'une procédure ancienne, due à Cooper [COO63] et qui peut être formulée comme suit :

```
0) Entrée : localisation des n entités et des p centres.
1) Répéter, jusqu'à ce que les allocations ne changent plus :
   1a) Allouer les n entités à leur centre le plus proche.
   1b) Localiser optimalement les centres relativement aux allocation faites au pas 1a.
```

Pour le problème de la p-médiane, la localisation optimale d'un centre peut se faire en essayant successivement toutes les entités rattachées à ce centre. Pour le problème de la minimisation de la somme des carrés des distances, le centre doit être placé au centre de gravité des entités qui lui sont allouées. Finalement, pour le problème de Weber, on peut utiliser une procédure de gradient [WEI37].

Le second voisinage, V_2, consiste à déplacer un centre sur une entité. Étant donné qu'il y a p centres et n entités, la taille de ce voisinage est pn; dans le cas de la p-médiane, il peut cependant être limité à $p(n-p)$ car il est inutile de déplacer un centre sur une entité qui en compte déjà un. Une fois que la solution a été perturbée par une modification selon V_2, on peut réappliquer la recherche locale pour trouver une solution stable et la conserver lorsqu'elle est la meilleure que l'algorithme a trouvé jusque là.

Cette simple procédure pseudo-gloutonne produit d'excellentes solutions. Dans le tableau 1, nous reportons, pour 40 problèmes de p-médiane dûs à Beasley [BEA85], la qualité moyenne des solutions qu'elle produit (mesurée en % au-dessus de l'optimum global) ; le tableau 2 donne la qualité de la plus mauvaise qu'elle a produite sur 20 exécutions partant de solutions initiales aléatoires et le tableau 3 donne la proportion de recherches ayant trouvé l'optimum global (estimé sur 20 essais).

$p \setminus n$	100	200	300	400	500	600	700	800	900
5	0	0	0	0.03	0	0	0.00	0.06	0.12
10	0.04	0.08	0	0.14	0.25	0	0.02	0.13	0
$n/10$	0.04	0.00	0.01	0.10	0.10	0.05	0.01	0.07	0.06
$n/5$	0.04	0.15	0.04	0.05	0.05	0.04	0.06	—	—
$n/3$	0.04	0.05	0.06	0.09	0.14	0.14	—	—	—

Tableau 1 : *Qualité de la procédure pseudo-gloutonne (% au-dessus de l'optimum).*

$p \setminus n$	100	200	300	400	500	600	700	800	900
5	0	0	0	0.26	0	0.07	0.01	1.16	0.40
10	0.29	0.32	0	0.54	1.01	0	0.04	0.55	0
$n/10$	0.87	0.07	0.11	0.29	0.21	0.16	0.09	0.26	0.21
$n/5$	0.39	0.51	0.13	0.17	0.24	0.20	0.23	—	—
$n/3$	0.22	0.56	0.17	0.39	0.49	0.30	—	—	—

Tableau 2 : *Qualité de la plus mauvaise des solution produites par la procédure pseudo-gloutonne sur 20 exécutions (% au-dessus de l'optimum).*

$p \setminus n$	100	200	300	400	500	600	700	800	900
5	1.0	1.0	1.0	0.9	1.0	0.7	0.6	0.95	0.7
10	0.85	0.5	1.0	0.3	0.75	1.0	0.55	0.55	1.0
$n/10$	0.95	0.95	0.95	0.2	0.3	0.4	0.55	0.2	0
$n/5$	0.9	0.2	0.45	0.3	0.7	0.4	0.2	—	—
$n/3$	0.8	0.85	0.25	0.4	0.25	0.05	—	—	—

Tableau 3 : *Proportion de recherches trouvant l'optimum global.*

Nous voyons que cette procédure trouve presque toujours l'optimum pour de faibles valeurs de p (5 et 10) mais que la proportion de recherches fructueuses diminue au fur et à mesure que p augmente ($p = n/10$, $p = n/5$, $p = n/3$). Cependant, la qualité moyenne des solutions est toujours très élevée, au pire 1/4% au-dessus de l'optimum. La plus mauvaise des solutions produites par cette procédure est à peine au-dessus de 1% et seules 3 exécutions sur 800 n'ont pas trouvé de solution à moins de 1%. On peut donc affirmer qu'elle est robuste.

Notons que la qualité des solutions stables (obtenue par la procédure de Cooper sur une solution initiale aléatoire) s'avère très mauvaise ; il est rare d'obtenir une solution à moins de 10% de l'optimum et la meilleure de 100 exécutions est rarement au-dessous de 1% mais peut monter jusqu'à près de 30% pour les plus grandes valeurs de p.

Nombre de centres	Meilleure solution connue (pseudo-glouton)	Meilleure, voisinage variable	Moyenne, voisinage variable	Meilleur optimum local / 1000	Moyenne, pseudo-glouton
10	1754840214	0.000	0.140	0.001	0
20	791794596.2	0.017	0.757	0.190	0.006
30	481251642.9	0.218	1.078	0.619	0.000
40	341342885.9	0.440	1.250	2.013	0.053
50	255723992.5	0.457	1.877	5.982	0.515
60	197273037.6	0.952	1.542	7.898	0.073
70	158450591.9	0.840	1.620	10.690	0.011
80	128890171.4	0.891	1.641	13.067	0.000
90	110456793.7	0.784	1.514	16.308	0.050
100	96330296.40	1.061	2.221	15.243	0.081
110	84849661.98	1.695	3.035	17.786	0.008
120	75545061.47	1.106	2.126	17.962	0.000
130	67561764.35	1.267	1.937	21.909	0.010
140	61128895.04	0.979	2.409	20.352	0.056
150	55918930.43	1.361	2.701	18.857	0.083
160	51313858.12	1.748	2.448	25.065	0.064

Tableau 4 : *Comparaison de la qualité des solutions obtenues avec les méthodes pseudo-gloutonne, à voisinage variable et optima locaux.*

Pour des problèmes de minimisation de la somme des carrés à 1060 entités et entre 10 et 160 centres, nous avons pu améliorer jusqu'à 1,75% la meilleure des solutions publiées, obtenue à l'aide d'une recherche à voisinage variable [HAN97], en effectuant seulement 3 exécutions de notre algorithme pseudo-aléatoire, toutes les solutions ayant été améliorées. Nous reportons dans le tableau 4 les résultats que nous avons obtenus avec notre algorithme pseudo-glouton et le comparons à la meilleure version de méthode à voisinage variables de [HAN97], ainsi qu'au meilleur optimum local obtenu avec la procédure de Cooper en partant de 1000 solutions initiales aléatoires. Nous voyons que les solutions moyennes obtenues par l'algorithme pseudo-glouton sont généralement meilleures que les meilleures des 10 solutions obtenues par la méthode à voisinage variable ; cette dernière pouvant donner des solutions en moyenne à 2 ou 3% au-dessus de la meilleure solution connue. Quant aux optima locaux, nous pouvons constater qu'ils se trouvent bien au-dessus de l'optimum global, parfois à plus de 25%.

Pour des problèmes de Weber, nous avons pu obtenir toutes les solutions optimales de 35 problèmes à 287 entités comprenant entre 2 et 100 centres (avec une marge d'erreur inférieure à 0,01% due à la précision des calculs), avec 5 répétitions de notre algorithme pseudo-glouton. Ces problèmes sont actuellement les plus grands pour lesquels des solutions exactes ont été publiées.

Il est donc surprenant de constater qu'une simple méthode de descente puisse obtenir des solutions de qualité si élevée, et qu'elle soit souvent meilleure que des recherches avec tabous [MLA96, VOß96]. Cependant, notre procédure pseudo-gloutonne est d'une complexité relativement élevée : la procédure de Cooper est d'une complexité d'au moins $O(pn)$ car on doit répéter un certain nombre de fois la procédure d'allocation des entités aux centres, ce qui demande un effort en $O(pn)$. En pratique cependant, on observe que la boucle 1 de cette procédure n'est répétée qu'un petit nombre de fois. Comme le voisinage V_2 est de taille $O(pn)$, on peut en déduire que la complexité totale de la procédure pseudo-gloutonne est approximativement $O(p^2n^2)$. Pour de grandes valeurs de p, cette complexité est excessive, mais il faut noter que la procédure trouve d'excellentes solutions bien avant qu'elle ait énuméré tout V_2.

Sur la base de cette constatation, nous avons proposé la méthode CLS (candidate list search) dans [TAI96b] pour accélérer la recherche. Elle consiste à choisir l'ordre dans lequel les éléments de V_2 sont énumérés, dans l'esprit des listes de mouvements candidats de Glover : au lieu d'essayer de déplacer un centre sur une entité quelconque, l'entité sur laquelle il est déplacé doit satisfaire à un critère d'éloignement du centre auquel elle était allouée. En modifiant à chaque essai de déplacement le centre et l'entité, on peut notablement accélérer l'obtention de bonnes solutions. Ainsi, sur des problèmes de Weber, on observe typiquement des solutions à une fraction de pourcent au-dessus de l'optimum pour des problèmes comportant plusieurs centaines d'entités et jusqu'à une trentaine de centres en effectuant que 100 à 200 itérations de la recherche pseudo-gloutonne.

Lorsque le nombre de centres augmente, la qualité de cette méthode se dégrade (pour un nombre fixé d'itérations) et il convient de limiter encore le nombre de voisins de V_2 énumérés. Pour cela, nous proposons la méthode LOPT, dans [TAI96b], qui consiste à considérer des sous-problèmes contenant $r < p$ centres, ainsi que les entités leur étant allouées, et de les optimiser indépendamment avec une recherche pseudo-gloutonne avec liste de mouvements candidats. Pour former un sous-problème, les r centres sont choisis comme suit : on élit tout d'abord un centre selon une certaine politique et les $r - 1$ autres sont ceux qui lui sont les plus proches. La politique du choix du premier centre est elle-même basée sur une liste de candidats. Initialement, tous les centres sont candidats. Lorsque l'on ne réussit pas à améliorer un sous-problème, le premier centre choisi est éliminé de la liste. Si au contraire on réussi à améliorer le sous-problème, les $r - 1$ autres centres y sont insérés pour autant qu'ils ne fassent pas déjà partie de la liste. Le processus s'arrête dès que la liste est vide.

Deux éléments sont indispensables pour que ce processus fonctionne bien : en premier lieu, il faut que les sous-problèmes puissent être résolus avec efficience et efficacité. Nous avons observé qu'il faut limiter le nombre r de centres d'un sous-problème à 10 ou 20 unités et effectuer entre 50 et 200 itérations de la recherche pseudo-gloutonne. En second lieu, il faut démarrer d'une solution initiale qui ait une bonne structure. Dans [TAI96b], nous proposons pour cela une procédure très rapide, DEC, basée sur la décomposition du problème et sa reconstruction optimale par programmation dynamique.

De la complexité initiale de l'algorithme pseudo-glouton qui était grosso modo en $O(n^4)$ lorsque $p = O(n)$, nous avons passé à une complexité empirique à peine supérieure à $O(n^2)$ pour LOPT, soit une complexité à peu près identique à la procédure de recherche d'une solution stable. Ces complexités ont été évaluées empiriquement sur plusieurs problèmes de classification avec centroïdes : en effet, le nombre d'itérations nécessaires pour trouver une solution stable dans la procédure de Cooper ou pour vider la liste de centres candidats dans LOPT dépend de la solution initiale et ne peut pas être formulé algébriquement, du moins dans l'état actuel des connaissances.

Comme la complexité de nos méthodes est très basse, nous avons pu attaquer des problèmes de plusieurs ordres de grandeur plus élevés que ceux que l'on peut trouver dans la littérature, jusqu'à plus de 85'000 entités et 15'000 centres.

3.3. Généralisation de LOPT

La procédure LOPT est facilement généralisable et peut s'appliquer à tout problème décomposable, dans la mesure où l'on peut attribuer une mesure pertinente de proximité entre les sous-problèmes générés. Étant donné une solution s d'un problème, solution composée de parties ou sous-problèmes $s_1, \ldots, s_p$ ($s = \bigcup_{i=1}^{p} s_i$, $\forall i \neq j \; s_i \cap s_j = \emptyset$) et une mesure de distance $d(s_i, s_j)$ entre sous-problèmes, LOPT peut se formuler comme suit :

Trame générale de LOPT

```
0)Entrée : Solution s composée de parties s1, …, sp.
1)C = {s1, …, sp}
2)Tant que C ≠∅, répéter :
```

2a) `Choisir` $s_i \in C$; `soit le sous-problème` R `com-posé de l'ensemble des` r `plus proches sous-problèmes de` s_i ($s_i \in R$)

2b) `Optimiser` R

2c) `Si` R `a été amélioré, poser` $C \leftarrow C \cup R$;
`sinon, poser` $C \leftarrow C \setminus \{s_i\}$.

Dans notre application au problèmes de regroupement avec centroïdes, le sous-problème s_i est composé de l'ensemble des entités allouées au centre i, la fonction de distance entre sous-problèmes est simplement la distances entre les centres qui les définissent et la procédure d'optimisation consiste à exécuter la procédure CLS pour un certain nombre d'itérations.

LOPT peut également généraliser la méthode pseudo-gloutonne que nous avons proposée pour les problèmes euclidiens de distribution dans [TAI93]. Dans ce cas, le sous-problème s_i est constitué des clients de la tournée i, la distance entre sous-problèmes est la distance entre les centres de gravité des clients qui les composent et la fonction d'optimisation consiste à appliquer une recherche avec tabou pour un certain nombre d'itérations. On peut également appliquer LOPT aux problèmes d'élaboration de tournées de véhicules comportant un grand nombre de dépôts selon un schéma un peu différent mais plus général : le sous-problème s_i est constitué des clients desservis à partir du dépôt i, la fonction de distance entre sous-problèmes étant simplement la distance entre dépôts.

On peut penser que LOPT devrait fonctionner avec satisfaction pour les problèmes où la fonction de distance permet d'isoler des parties de solutions fortement dépendantes (pour la création des sous-problèmes à optimiser) ou indépendantes (pour éliminer les sous-problèmes constituées de parties déjà résolues optimalement). Ces caractéristiques sont clairement remplies pour les problèmes d'élaboration de tournées de véhicules et dans le domaine de la classification avec centroïdes que nous avons abordés dans [TAI93] et [TAI96b] et c'est certainement ce qui fait que LOPT fonctionne bien dans ces cas-là. Cependant, son application à d'autres problèmes reste à évaluer.

4. Calcul Concurrent

Une des particularités de la programmation à mémoire adaptative ou des méthodes pseudo-gloutonnes est leur bon potentiel d'exécution concurrente, moyennant quelques adaptations algorithmiques mineures relativement aux principes de fonctionnement généraux qui ont été décrits ci-dessus. En effet, il suffit de constater que la sous-procédure de recherche locale ou d'optimisation contenue dans ces algorithmes est la partie la plus gourmande en temps de calcul et qu'il est relativement facile de l'isoler dans des processus concurrents.

4.1. Programmation à mémoire adaptative

Bien que nous n'ayons pas formulé l'algorithme général de la programmation à mémoire adaptative sous une forme concurrente, il est cependant facile de le paralléliser. En effet, il suffit pour cela d'exécuter la boucle principale de manière indépendante grâce à plusieurs processus qui ne communiquent entre eux que par l'intermédiaire de la mémoire. Cela signifie qu'une machine idéale pour l'exécution concurrente d'un programme à mémoire adaptative doit posséder une mémoire partagée.

4.1.1. Machine à mémoire commune

Le schéma d'une parallélisation sur une machine disposant d'une mémoire commune est présenté en figure 1. Ce schéma est généralement très efficace car le processus de lecture de la mémoire et de sa mise à jour est beaucoup plus rapide que les autres, en particulier que celui de la recherche locale qui consomme typiquement presque l'intégralité des ressources de calcul. On suppose cependant que chaque processus d'optimisation possède une copie locale des données du problème.

Nous pouvons illustrer une implantation typique d'un programme à mémoire adaptative parallèle sur le problème de l'affectation quadratique. La solution d'un tel problème, pour un exemple de taille n, peut être représentée par une permutation de n

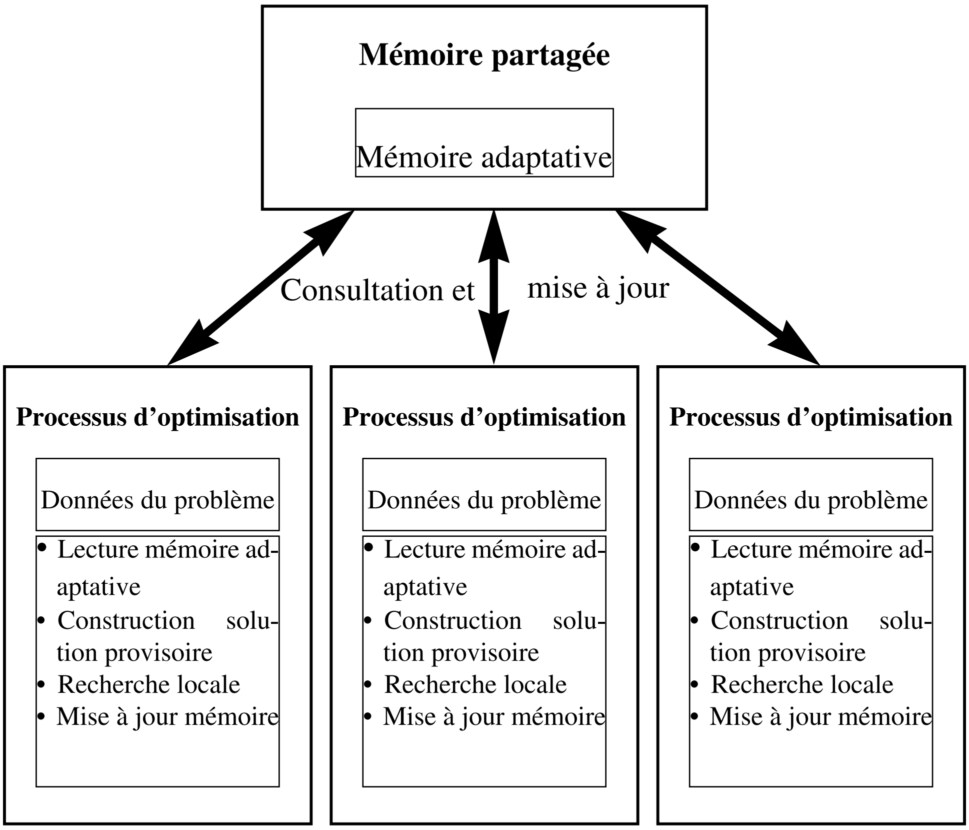

Figure 1 : *Parallélisation avec une mémoire commune.*

éléments. Les données du problème sont constituées de deux matrices carrées de dimension $n{\times}n$. Dans nos récentes implantations [GAM97, TAI97b], de même que dans l'implantation de [FLE94], la mémoire adaptative est elle aussi constituée d'une matrice carrée de dimension $n{\times}n$ (ou éventuellement de $O(n)$ permutations de n éléments, ce qui correspond à un encombrement similaire). Cela signifie que la quantité d'informations transférées pour consulter la mémoire ou la mettre à jour est au pire en $O(n^2)$, mais souvent en $O(n)$, le temps nécessaire pour calculer les nouvelles valeurs de la mémoire étant du même ordre de complexité. Par contre, le processus de recherche locale est d'une complexité supérieure : $O(n^3)$. Ainsi, des problèmes de congestion dans les accès à la mémoire partagée pourront apparaître pour un nombre de processus d'optimisation de l'ordre de $O(n)$, voire $O(n^2)$.

4.1.2. Machines sans mémoire commune

Les machines à mémoire commune ne sont cependant pas très courantes ; un réseau de stations de travail est l'exemple type de machine parallèle utilisée aujourd'hui. Pour paralléliser un programme à mémoire adaptative sur un tel réseau, nous avons défini un processus chargé de simuler une mémoire partagée en plus des processus de recherche locale. Comme les opérations spécifiques à réaliser sur la mémoire ne sont pas exigeantes en temps de calcul, il est possible de les faire au moyen du processus simulant la mémoire. On arrive ainsi au schéma de parallélisation présenté dans la figure 2.

Il faut également noter que la recherche locale elle-même peut être concurrente. Dans ce cas, le processus de gestion de la mémoire peut aussi englober un processus

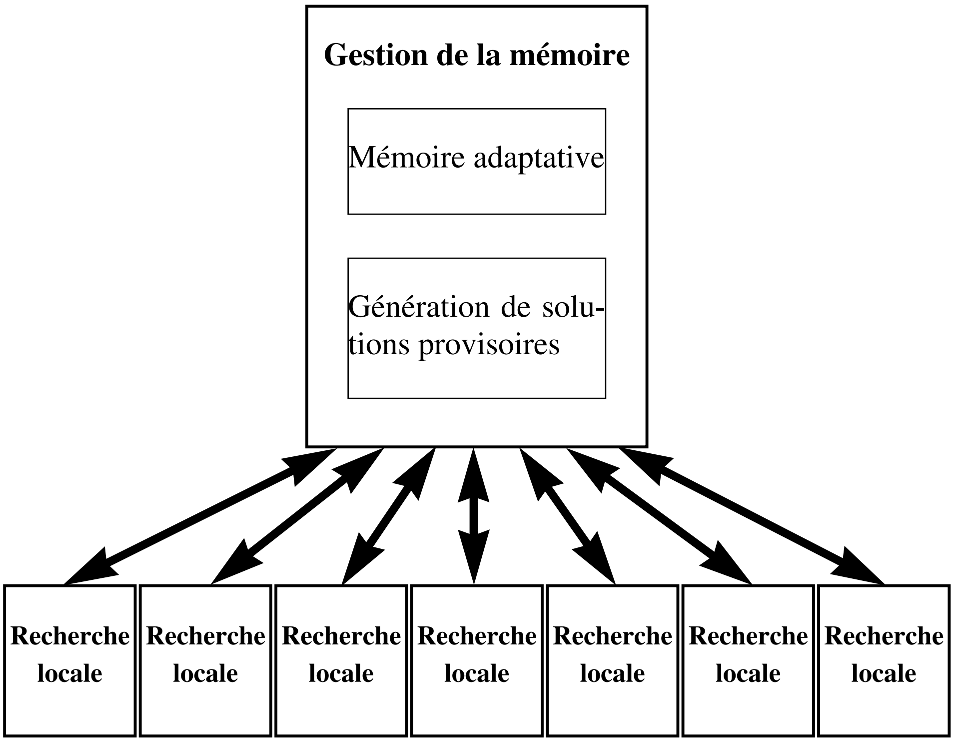

Figure 2 : *Parallélisation d'un programme à mémoire adaptative par processus communicants.*

d'équilibrage des charges, comme nous l'avons implanté dans Badeau et al. [BAD97]. De cette manière, nous avons pu observer des efficacités de parallélisation allant de 60 à 75% pour un nombre de processeurs compris entre 4 et 17 et pour une application à un problème de distribution de biens. Cette première mesure de l'efficacité correspond au rapport entre le temps de calcul sur une machine séquentielle et le temps sur la machine parallèle multiplié par le nombre de ses processeurs. On pourrait définir une seconde mesure, en prenant en considération le temps nécessaire pour obtenir des solutions de qualité donnée sur les deux types de machine. En effet, les deux algorithmes, séquentiels ou parallèles, ne fonctionnent pas de la même manière : l'algorithme séquentiel construit la solution provisoire sur la base de l'ensemble de toutes les solutions trouvées par le programme, alors que le processus de génération de solutions provisoires de l'algorithme parallèle ignore les solutions en cours de traitement par les autres processus. L'algorithme séquentiel met donc à jour la mémoire progressivement, alors que l'algorithme parallèle fait une mise à jour par paquet. Nous avons cependant remarqué que pour un faible nombre de processeurs (jusqu'à 16), la qualité des solutions obtenues dépendait essentiellement de la quantité totale de travail réalisée, indépendamment du nombre de processeurs utilisés pour réaliser ce travail [BAD97]. D'un point de vue pratique, les deux mesures d'efficacité sont donc plus ou moins équivalentes.

4.2. Méthodes pseudo-gloutonnes

Une méthode pseudo-gloutonne est composée en fait de deux boucles imbriquées, la plus extérieure consistant à effectuer des sauts ou à définir des sous-problèmes à l'aide d'un voisinage V_2, alors que la boucle intérieure réalise une recherche locale avec un voisinage V_1. Comme cette recherche locale est souvent une partie coûteuse en temps de calcul, il est naturel de tenter de paralléliser une recherche pseudo-gloutonne en ayant recours à des processus indépendants pour les recherches locales dans V_1.

Dans le cadre d'un voisinage V_2 définissant des sous-problèmes, une voie de parallélisation consiste à choisir des sous-problèmes indépendants pouvant être optimisés indépendamment les uns des autres, comme nous l'avons proposé dans [TAI93, BAD97] pour les problèmes de distribution. Cette technique peut également être appliquée à nos méthodes LOPT et DEC pour la classification avec centroïdes. Cependant, elle n'a été implantée sur une machine concurrente que pour des problèmes de distribution. Pour d'autres applications, un bon nombre d'inconnues doivent être levées afin d'obtenir une efficacité acceptable, comme par exemple celle de la politique de génération d'un nombre aussi élevé que possible de sous-problèmes indépendants, tout en couvrant l'ensemble de V_2. En effet, on remarquera que V_2 n'est pas forcément statique : par exemple, pour LOPT, une itération peut éliminer un sous-problème ou en créer de nouveaux.

Lorsqu'il n'est pas possible de définir des sous-problèmes indépendants, ou lorsque le voisinage V_2 définit des sauts dans l'espace des solutions par rapport au voisinage limité V_1, la parallélisation de l'algorithme devient plus problématique. En effet, en début de recherche, la fréquence d'amélioration des solutions est élevée, et une parallélisation efficace devra vraisemblablement modifier l'algorithme de la politique « première amélioration trouvée» pour celle, souvent beaucoup plus lente en séquentiel « meilleure amélioration possible ». Le potentiel de parallélisation est toutefois très élevé car on peut penser effectuer jusqu'à $|V_2|$ processus de recherches locales en parallèle. Cette voie de recherche mérite sans nul doute d'être explorée.

5. Conclusions

5.1. Synthèse des contributions

Les contributions dans le domaine de l'optimisation combinatoire apportées par ce travail sont multiples. Tout d'abord, nous nous sommes penché sur les fondements de diverses techniques générales d'optimisation et nous avons montré que les méta-heuristiques à mémoire se sont rapprochées les unes des autres ces dernières années, au point de se rencontrer parfois. La dénomination de *programmation à mémoire adaptative* nous semble à la fois plus précise et plus générale car elle est mieux représentative du fonctionnement des dernières évolutions des méta-heuristiques et laisse cependant une grande liberté dans le choix de l'implantation.

Ensuite, les résultats numériques que nous avons obtenus dans divers domaines de l'optimisation combinatoire nous permettent de dire que la programmation à mémoire adaptative semble une des techniques les plus prometteuses à l'heure actuelle, et cela pour plusieurs raisons.

- Elle permet d'obtenir des solutions de qualité élevée.
- Elle reste conceptuellement simple et donc relativement facilement implantable.
- Elle est facilement parallélisable.
- Elle est efficace et particulièrement bien adaptée aux besoins des applications pratiques car on peut l'implanter de telle sorte qu'il soit possible d'obtenir n'importe quand durant la recherche non pas une seule, mais plusieurs bonnes solutions simultanément, ce qui est un atout non négligeable pour les applications réelles.
- Elle est bien adaptée à la résolution de problèmes dynamiques.

Ces deux derniers points mériteront une attention particulière dans un futur relativement proche. En effet, si les méta-heuristiques n'ont encore fait que de timides apparitions dans les logiciels commerciaux, il n'en reste pas moins qu'elles y sont de plus en plus présentes, l'augmentation de la puissance des micro-ordinateurs n'étant certainement pas étrangère à ce phénomène. Avec la perspective de pouvoir les utiliser pour des applications dynamiques, sur un réseau de calculateurs, on peut penser que la programmation à mémoire adaptative a de beaux jours devant elle.

Finalement, dans le chapitre traitant de la recherche pseudo-gloutonne, nous avons montré que de simples méthodes de descente pouvaient être très compétitives, tant du point de vue de la qualité des solutions trouvées que du temps de calcul. Ceci peut paraître surprenant si l'on considère l'importante littérature concernant les recherches locales. Force est de constater que la recherche pseudo-gloutonne doit donc être envisagée comme une alternative sérieuse face à d'autres méta-heuristiques comme la recherche avec tabous ou les algorithmes génétiques.

5.2. Perspectives de recherche

Les bases de la programmation à mémoire adaptative posées dans ce travail sont susceptibles de développer de nouvelles voies de recherches. Tout d'abord, mentionnons la parallélisation de cette technique. En effet, nous avons implanté la plupart de nos méthodes sur une machine séquentielle et nous nous sommes contenté de montrer que même sur une telle machine, elles étaient parfaitement compétitives. Nous n'avons que timidement montré qu'elles pouvaient se paralléliser efficacement pour un faible nombre de processeurs. Il reste donc à évaluer et à étudier leur implantation sur des machines dotées de centaines ou de milliers de processeurs.

Ensuite, une autre voie de recherche qui mérite d'être développée concerne les problèmes dynamiques, d'une part parce que ce type de problème n'a encore été que très peu abordé dans la littérature alors que la plupart des problèmes évoluent avec le temps dans la réalité, et d'autre part parce qu'un programme à mémoire adaptative doit certainement mettre à jour sa mémoire sur des bases quelque peu différentes dans le cas de problèmes statiques ou dynamiques. Dans ce dernier cas, il faut certainement tenir compte de la robustesse des solutions trouvées. En effet, une excellente solution peut devenir irréalisable si les données sont très légèrement modifiées. On préférera donc en pratique une solution théoriquement moins bonne mais aussi moins contraintes et plus facilement adaptable à un monde dynamique ou incertain.

Finalement, les méthodes pseudo-gloutonnes, et en particulier la simple méthode générale d'optimisation de sous-problèmes LOPT méritent un étude plus approfondie. En premier lieu, on peut penser à utiliser une procédure exacte d'optimisation des sous-problèmes à la place des méthodes heuristiques incorporées dans nos implantations. Cela permettrait de mieux évaluer à la fois les limites de LOPT, puisqu'il n'y aurait plus d'incertitude sur la qualité de la résolution des sous-problèmes, ainsi que de mettre au point des méthodes exactes robustes. En effet, il n'est pas forcément possible d'incorporer directement n'importe quel algorithme exact dans LOPT, même s'il s'agit d'un « champion » dans sa catégorie : une mesure classique de la puissance d'un algorithme exact est la taille du plus grand problème qu'il a résolu. Or, la résolution de quelques exemples de problèmes de grande taille n'implique pas forcément une résolution rapide pour tout exemple de petite taille. Comme LOPT requiert la résolution d'un grand nombre de sous-problèmes, la probabilité d'avoir à résoudre un problème difficile bien que de petite taille augmente fortement.

Deuxièmement, les performances théoriques de LOPT pourraient être estimées pour certains problèmes d'optimisation combinatoire. Même si l'on ne connaît toujours pas à l'heure actuelle la performance théorique de beaucoup de procédures de recherche d'optima locaux relativement à des voisinages simples, il n'est pas interdit de penser qu'une telle estimation puisse se faire plus aisément pour LOPT qui travaille de manière plus globale.

Troisièmement, la programmation concurrente des méthodes pseudo-gloutonnes reste encore à étudier. Il est indéniable qu'elles présentent de bons potentiels de parallélisation. Par exemple, dans une implantation synchrone, LOPT pourrait utiliser jusqu'à p processus d'optimisation à la fois, où p est le nombre de sous-problèmes que l'on crée à partir d'une solution de départ. Ces derniers n'étant pas indépendants les uns des autres, il faut donc trouver de bonnes politiques pour décider quels sous-problème optimisé sera retenu d'une synchronisation à la suivante. Une implantation asynchrone ne manquerait pas de soulever d'autres difficultés qu'il faudrait aplanir.

6. Bibliographie

[BAD97] P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin & É. D. Taillard, « A parallel tabu search heuristic for the vehicle routing problem with time windows », *Transportation Research-C 5*, 1997, pp. 109–122.

[BAT94] R. Battiti & G. Tecchiolli, « The reactive tabu search », *ORSA Journal on Computing 6*, 1994, pp. 126–140.

[BEA85] J. E. Beasley, « A note on solving large *p*-median problems », *European Journal of Operational Research 21*, 1985, pp. 270–273.

[COL91] A. Colorni, M. Dorigo & V. Maniezzo, « Distributed Optimization by Ant Colonies », actes de *ECAL91 — European Conference on Artificial Life*, 1991, pp. 134–142.

[COO63] L. Cooper, « Location-allocation problems », *Operations Research 11*, 1963, pp. 331–343.

[CUN97] V.-D. Cung, T. Mautor, Ph. Michelon & A. Tavares, « A Scatter Search Based Approach for the Quadratic Assingment Problem », Rapport technique, Université de Versailles-Saint Quentin en Yvelines, 1997.

[DOR96] M. Dorigo, V. Maniezzo & A. Colorni, « The Ant System : Optimization by a colony of cooperating agents », *IEEE Transactions on Systems, Man, an Cybernetics-Part B 26*, 1996, pp. 29–41.

[DOR97] M. Dorigo & L. M. Gambardella, « Ant colony system : A cooperative Learning Approach to the Traveling Salesman Problem », *IEEE Transactions on Evolutionary Computing 1*, 1997, pp. 53–66.

[ELS77] A. E. Elshafei, « Hospital Layout as a Quadratic Assignment Problem », *Operations Research Quaterly 28*, 1977, pp. 167–179.

[FAI92] U. Faigle & W. Kern, « Some convergence results for probabilistic tabu search », *ORSA Journal on Computing 4*, 1992, pp. 32–37.

[FLE94] C. Fleurent & J. Ferland, « Genetic hybrids for the quadratic assignment problem », *DIMACS Series in Mathematics and Theoretical Computer Science 16*, 1994, pp. 190–206.

[FLE96] C. Fleurent, F. Glover, P. Michelon & Z. Valli, « A Scatter Search Approach for Unconstrained Continuous Optmization », actes de *IEEE International Conference on Evolutionary Computation*, 1996, pp. 643–648.

[GAM97] L. M. Gambardella, É. D. Taillard & M. Dorigo, « Ant colonies for the quadratic assignment problem », rapport technique *IDSIA-4-97*, IDSIA, Lugano, 1997.

[GEN94] M. Gendreau, A. Hertz & G. Laporte, « A tabu search heuristic for the vehicle routing problem », *Management Science 40*, 1994, pp. 1276–1290.

[GEN96a] M. Gendreau, P. Badeau, F. Guertin, J.-Y. Potvin & É. D. Taillard, « A solution procedure for real-time routing and dispatching of commercial vehicles », actes du *3. World Congress on Intelligent Transport Systems*, Orlando (Floride), 1996.

[GEN96b] M. Gendreau, F. Guertin, J.-Y. Potvin & É. D. Taillard, « Tabu search for real-time vehicle routing and dispatching », rapport technique *CRT-96-47*, Centre de recherche sur les transports, Université de Montréal, 1996.

[GLO77] F. Glover, « Heuristics for Integer Programming Using Surrogate Constraints », *Decision Sciences 8*, 1977, pp. 156–166.

[GLO86] F. Glover, « Future Paths for Integer Programming and Links to Artificial Intelligence », *Computers and Operations Research 13*, 1986, pp. 156–166.

[GLO89] F. Glover, « Tabu Search — Part I », *ORSA Journal on Computing 1*, 1989, pp. 190–206.

[GLO90] F. Glover, « Tabu Search — Part II », *ORSA Journal on Computing 2*, 1990, pp. 4–32.

[GLO97a] F. Glover « A template for Scatter Search and Path Relinking » rapport technique, Université du Colorado à Boulder, 1997.

[GLO97b] F. Glover « Tabu Search and adaptive memory programming — advances, applications and challenges », dans : *Interfaces in Computer Science and Operations Research*, Barr, Helgason and Kennington éditeurs, Kluwer Academic Publishers, 1997, pp. 1–75.

[GLO97c] F. Glover & M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.

[GOL97] B. L. Golden, G. Laporte, É. D. Taillard, « An adaptive memory heuristic for a class of vehicle routing problems with minmax objective » , *Computers and OR 24*, 1997, pp. 445–452.

[HAN97] P. Hansen & N. Mladenovic, « An Introduction to Variable neighbourhood Search for the *p*-Median », conférence *MIC 97*, Sophia-Antipolis, juillet 1997.

[HAJ88] B. Hajek, « Cooling schedules for optimal annealing » *Mathematics of Operations Research 13*, 1988, pp. 311–329.

[HOL75] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press : Ann Arbor, 1975.

[KAR79] O. Kariv & S. L. Hakimi, « An algorithmic approach to network location problems, Part 2 : the p-medians », *SIAM Journal of Applied Mathematics 37*, 1979, pp. 539–560.

[KEL96] J. Kelly, B. Rangaswamy & J. Xu, « A Scatter Search-Based Learning Algorithm for Neural Network Training », *Journal of Heuristics 2*, 1996, pp. 129–146.

[KOO57] T. C. Koopmans & M. J. Beckmann, « Assignment problems and the location of economics activities », *Econometrica 25*, 1957, pp. 53–76.

[LAP88] G. Laporte & H. Mercure, « Balancing Hydraulic Turbine Runners : A Quadratic Assignment Problem », *European Journal of Operational Research 35*, 1988, pp. 378–381.

[LIN73] S. Lin & B. W. Kernighan, « An effective heuristic algorithm for the traveling salesman problem », *Operations Research 21*, 1973, pp. 498–516.

[MLA95] N. Mladenovic, « A Variable Neighbourhood Algorithm », résumés des *journées de l'optimisation*, Montréal, Canada, 1995, p. 112.

[MLA96] N. Mladenovic, J.P. Moreno & J. Moreno-Vega, « A Chain-Interchange Heuristic Method », *Yugoslav Journal on Operations Research 6*, 1996, pp. 41–54.

[MÜL88] H. Mühlenbein, M. Gorges-Schleuter & O. Krämer, « Evolution Algorithms in Combinatorial Optimization », *Parallel Computing 7*, 1988, pp. 65–88.

[NOW96] E. Nowicki & C. Smutnicki, « A fast taboo search algorithm for the job shop problem » *Management Science 42*, 1996, pp.797–813.

[ROC94] Y. Rochat & F. Semet, « A Tabu Search Approach for Delivering Pet Food an Flour in Switzerland », *Journal of the Operations Research Society 45*, 1994, pp. 1233–1246.

[ROC95] Y. Rochat & É. D. Taillard, « Probabilistic diversification and intensification in local search for vehicle routing ». *Journal of Heuristics 1*, 1995, pp. 147–167.

[ROL97] E. Roland, D. A. Schilling & J. R. Current, « An Efficient tabu search Procedure for the p-median Problem ». *European Journal of Operational Research 96*, 1997, pp. 329–342.

[SHA76] S. Shani & T. Gonzalez, « P-complete approximation problems », *Journal of the ACM 23*, 1976, pp. 555–565.

[STÜ97] T. Stützle & H. Hoos, « MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems — Towards Adaptive Tools for Combinatorial Global Optimization », conférence *MIC 97*, Sophia-Antipolis, juillet 1997.

[TAI91] É. D. Taillard, « Robust taboo search for the quadratic assignment problem », *Parallel computing 17*, 1991, pp. 443–455.

[TAI93] É. D. Taillard, « Parallel iterative search methods for vehicle routing problems », *Networks 23*, 1993, pp. 661–673.

[TAI94] É. D. Taillard, « Parallel taboo search techniques for the job shop scheduling problem », *ORSA journal on Computing 6*, 1994, pp. 108–117.

[TAI95] É. D. Taillard, « Comparison of iterative searches for the quadratic assignment problem », *Location Science 3*, 1995, pp. 87–105.

[TAI96a] É. D. Taillard, « A heuristic column generation method for the heterogeneous VRP », rapport technique *CRT–96–03*, Centre de recherche sur les transports, Université de Montréal, 1996, à paraître dans *RAIRO–OR*

[TAI96b] É. D. Taillard, « Heuristic methods for large centroid clustering problems », Rapport technique *IDSIA–96–96*, IDSIA, Lugano, 1996.

[TAI96c] É. D. Taillard, G. Laporte & M. Gendreau, « Vehicle routeing with multiple use of vehicles », *Journal of the Operations Research Society 47*, 1996, pp. 1065–1070.

[TAI97a] É. D. Taillard, P. Badeau, M. Gendreau, F. Guertin & J.Y. Potvin, « A tabu search heuristic for the vehicle routing problem with soft time windows », *Transportation Science 31*, 1997, pp. 170–186.

[TAI97b] É. D. Taillard & L. M. Gambardella, « Adaptive Memories for the Quadratic Assignment Problem », rapport technique *IDSIA–87–97*, IDSIA, Lugano, 1997.

[TAT95] D. E. Tate & A. E. Smith, « A genetic approach to the quadratic assignment problem », *Computers and Operations Research 1*, 1995, pp. 855–865.

[VOß96] S. Voß, « A Reverse Elimination Approach for the *p*-Median Problem », *Studies in Locational Analysis 8*, 1996, pp. 49–58.

[WEI37] E. Weiszfeld, « Sur le point pour lequel la somme des distances de *n* points donnés est minimum », *Tôhoku Mathematical Journal 43*, 1937, 355–386.

ANNEXE I : CURRICULUM VITÆ

NOM : Taillard **PRÉNOMS** : Éric Denis

DATE DE NAISSANCE : 13 janvier 1965

NATIONALITÉ : Suisse

ADRESSE PROFESSIONNELLE :

IDSIA (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale), Corso Elvezia 36, 6900 Lugano, Tél. (091) 911 98 33, Fax. (091) 911 98 39,
courrier électronique : eric@idsia.ch

ADRESSE PRIVÉE : 6945 Origlio, Tél et fax. (091) 945 00 68.

SITUATION ACTUELLE : Chercheur et directeur de recherche.

FORMATION :

De 83 à 84 : École Polytechnique Fédérale de Lausanne (EPFL), Physique.
De 84 à 88 : EPFL, Informatique, orientation *logiciels d'application*.
En 88 : EPFL, Cours postgrade en informatique technique.
De 88 à 93 : EPFL, Thèse de doctorat en recherche opérationnelle.

DIPLÔMES ET / OU TITRES OBTENUS :

1988 Diplôme d'ingénieur informaticien EPFL.
1988 Certificat d'études postgrades en informatique technique, EPFL.
1993 Doctorat ès sciences, EPFL, Département de mathématiques.

CARRIÈRE PROFESSIONNELLE :

1988 –1994 : **Assistant-chercheur, EPFL**.

Chaire de recherche opérationnelle, Professeur D. de Werra, Département de mathématiques. Activité de recherche : Projet du Fonds National dans le domaine de l'algorithmique parallèle. Assistanat : Cours de premier cycle *Recherche opérationnelle*, cours de second cycle *Graphe et réseaux*.

1990–1994 : **Chargé de cours, EPFL.**

1994 - 1996 : **Chercheur**

Centre de recherche sur les transports, Université de Montréal, Canada. Lauréat d'une bourse internationale de recherche du Conseil de Recherche pour les Sciences Naturelles et du Génie du Canada.

Depuis mars 1996 : **Chercheur et directeur de recherche**

IDSIA, Lugano. Activités principales :

- Direction de doctorants travaillant à l'IDSIA.
- Projet du Fonds National pour la recherche scientifique : étude des méta-heuristiques basée sur le comportement des fourmis.
- Projet de modules d'optimisation pour la gestion des conteneurs d'un des plus grands ports au monde spécialisé dans le transport intermodal de conteneurs.
- Projet de gestion de la livraison de combustibles de la plus grande entreprise tessinoise dans le domaine.

ENSEIGNEMENT :

1990	*Modèles quantitatifs pour l'aide à la décision.* Cycle postgrade en environnement, Département de génie rural, EPFL.
90 - 91 et 92 - 93	*Optimisation.* Cours de second cycle, Départements de mathématique et d'informatique, EPFL.
1991	*Parallélisation de recherches itératives.* Cours postgrade : le parallélisme, Département d'informatique, EPFL.

1993	*Complexité et recherches itératives.* 1^er^ Cours postgrade en recherche opérationnelle, EPFL, Institut National Polytechnique et Université Joseph Fourrier de Grenoble.
93 - 94	*Compléments de recherche opérationnelle.* Département d'électricité, EPFL.
92–	Séminaires de 2^e^ et 3^e^ cycles dans diverses Hautes Écoles européennes.

ENCADREMENT DE RECHERCHE :

1988–1994 :	Supervision de nombreux travaux pratiques de semestre, diplôme et 3e cycle à l'EPFL.
1995 :	Supervision du Diplôme d'Études Approfondies de Ph. Badeau (Informatique des Systèmes de Production, Université Blaise Pascal Clermont-Ferrand II).
1996	Supervision de la thèse de maîtrise de Ch. Musaraganyi (Centre de recherche sur les transports, Univeristé de Montréal).
1996	Codirection de la thèse de lauréat de L. Mazzone et L. Romanelli (Département d'électronique et informatique, École polytechnique de Milan)
1997– :	Codirection des thèses de doctorat de M. Mastrolilli et G. Agazzi (IDSIA).

CONNAISSANCES LINGUISTIQUES :

Langue maternelle : Français.

Bonnes connaissances : Anglais, allemand, italien.

LISTE DES PUBLICATIONS

M. Gendreau, G. Laporte, C. Musaraganyi, É. D. Taillard, « A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem », Rapport technique IDSIA-22-98, 1998.

É. D. Taillard, « FANT: Fast ant system », Rapport technique IDSIA-46-98, IDSIA, Lugano, 1998

É. D. Taillard, L. M. Gambardella, M. Gendreau, J.-Y. Potvin, « Adaptive memory programming a unified view of meta-heuristics », Rapport technique IDSIA-19-98, IDSIA, Lugano, 1998.

É. D. Taillard, L. M. Gambardella, M. Gendreau, J.-Y. Potvin, « Programmation à mémoire adaptative », Calculateurs parallèles, Réseaux et Systèmes répartis 10, 1998, 117–140.

P. Badeau, F. Guertin, M. Gendreau, J.-Y. Potvin, É. D. Taillard, « A parallel tabu search heuristic for the vehicle routing problem with time windows », *Transportation Research–C 5*, 1997, 109-122.

J. Brimberg, P. Hansen, N. Mladenovic, É. D. Taillard, « A comparison of recent heuristic methods for solving the multisource Weber problem », Rapport technique IDSIA-33-97, IDSIA, Lugano, 1997.

L. M. Gambardella, É. D. Taillard, M. Dorigo, « Ant colonies for the quadratic assignment problem », Rapport technique IDSIA-4-97, IDSIA, Lugano, 1997.

B. L. Golden, G. Laporte, É. D. Taillard, « An adaptive memory heuristic for a class of vehicle routing problems with minmax objective », *Computers and OR 24*, 1997, 445-452.

A. Hertz, É. D. Taillard, D. de Werra, Chapitre « Tabu Search », E. Aarts et J. K. Lenstra (éditeur), *Local Search in Combinatorial Optimization*, Wiley, Chichester, 1997, 121-136.

É. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, « A tabu search heuristic for the vehicle routing problem with soft time windows », *Transportation Science 31*, 1997, 170-186.

É. D. Taillard, L. M. Gambardella, « An ant approach for structured quadratic assignment problems », Résumé étendu IDSIA-22-97, IDSIA, Lugano, 1997.

É. D. Taillard, L. M. Gambardella, « Adaptive memories for the quadratic assignment problem », Rapport technique IDSIA-97-97, IDSIA, Lugano, 1997.

L. M. Gambardella, G. Bontempi, É. D. Taillard, D. Romanengo, G. Raso, P. Piermari, « Simulation and forecasting in intermodal container terminal », Actes du 8e *European Simulation Symposium*, Genova, Italie, 24-26 octobre 1996.

M. Gendreau, F. Guertin, J.-Y. Potvin, É. D. Taillard, « Tabu search for real-time vehicle routing and dispatching », Publication *CRT-96-47*, Centre de recherche sur les transports, Université de Montréal, 1996.

M. Gendreau, P. Badeau, F. Guertin, J.-Y. Potvin, É. D. Taillard, « A solution procedure for real-time routing and dispatching of commercial vehicles », Publication *CRT-96-24*, Centre de recherche sur les transports, Université de Montréal, 1996.

P. Hansen, N. Mladenovic, É. D. Taillard, « Heuristic solution of the multisource Weber problem as a p-median problem », Publication du *GERAD*, Université de Montréal, 1996. À paraître dans *OR Letters*.

É. D. Taillard, « Heuristic methods for large centroid clustering problems », Rapport technique IDSIA-96-96, IDSIA, Lugano, 1996.

É. D. Taillard, « A heuristic column generation method for the heterogeneous VRP », Publication *CRT-96-03*, Centre de recherche sur les transports, Université de Montréal, 1996. À paraître dans *RAIRO-OR*.

É. D. Taillard, G. Laporte, M. Gendreau, « Vehicle routing with multiple use of vehicles », *Journal of the Operational Research Society 47,* 1996, 1065-1070.

Y. Rochat, É. D. Taillard, « Probabilistic diversification and intensification in local search for vehicle routing », *Journal of Heuristics 1*, 1995, 147-167.

É. D. Taillard, « Comparison of iterative searches for the quadratic assignment problem », *Location science 3*, 1995, 87-105.

É. D. Taillard, « Parallel taboo search techniques for the job shop scheduling problem », *ORSA Journal on Computing 6*, 1994, 108-117.

F. Glover, É. D. Taillard, D. de Werra, « A user's guide to taboo search », *Annals of Operations research 41*, 1993, 3-28.

F. Semet, É. D. Taillard, « Solving real-life vehicle routing problems efficiently using taboo search », *Annals of Operations research 41*, 1993, 469-488.

É. D. Taillard, « Parallel iterative search methods for vehicle routing problems », *Networks 23*, 1993, 661-673.

É. D. Taillard, *Recherches itératives dirigées parallèles*, Thèse de doctorat, Département de mathématiques, EPFL, 1993.

É. D. Taillard, « Benchmarks for basic scheduling problems », *European Journal of Operational Research 64*, 1993, 278-285.

É. D. Taillard, « Robust taboo search for the quadratic assignment problem », *Parallel Computing 17*, 1991, 443-455.

É. D. Taillard, « Some efficient heuristic methods for the flow shop sequencing problem », *European Journal of Operational Research 47*, 1990, 65-74.

PARTICIPATION À DES CONFÉRENCES ET SÉMINAIRES.

Membre du comité scientifique de *PAREO'98*, première rencontre du groupe de travail EURO « Parallel Processing in Operations Research », Versailles, France, juillet 8-10, 1998.

Membre du comité du programme de *ANTS'98 - From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimization*, Bruxelles, Belgique, octobre 15-16, 1998.

« Adaptive memory programming », session semi-plénière, conférence *EURO XVI*, Bruxelles, Belgique, juillet 1998.

Organisateur de quatre sessions « Meta-heuristics » à *EURO XVI*, Bruxelles, Belgique, juillet 1998.

« Adaptive memory programming for combinatorial problems », Séminaire à l'Université de Saint-Gall, Suisse, janvier 1998.

« Adaptive memory programming for various vehicle routeing problems », Séminaire à l'Université de Bologne, Italie, décembre 1997.

« Adaptive memory programming », Séminaire à Reggio en Émilie, Italie, décembre 1997.

« Ant hybrid for the quadratic assignment problem », *16. International Symposium on Mathematical Programming*, Lausanne, Suisse, Août 1997.

« Adaptive memories for the quadratic assignment problem », *Metaheuristic International Conference II*, Sophia-Antipolis, France, juillet 1997.

« Adaptive memories for the quadratic assignment problem », *Metaheuristic International Conference II,* Sophia-Antipolis, France, juillet 1997.

« Heuristic methods for large multi-source Weber problems », *EURO XV / INFORMS XXXIV*, Barcelone, Espagne, juillet 1997.

« Old (and new) heuristic methods for the multisource Weber problem », *Graphs and Optimization Colloquium III*, Loèche-les-Bains, Suisse, août 1996.

« Adaptive memory programming for vehicle routing problems », *Stratagem'96 Colloquium*, Sophia-Antipolis, France, juillet 1996.

« An introduction to taboo search », Séminaire donné au *GMD*, St. Augustin, Allemagne, juin 1996.

« Advanced taboo search or genetic hybrids », Séminaire donné au *GMD*, St. Augustin, Allemagne, juin 1996.

Organisateur de la session « Heuristics », *Optimization Days*, Montréal, Canada, mai 1996.

« A parallel adaptive memory procedure for the vehicle routing with soft time windows », *Parallel Optimization Colloquium*, Versailles, France, mars 1996.

« Diversification/intensification in local search for the vehicle routing problem », *Meeting of the Institute for Operations Research and the Management Sciences*, New Orleans, États-Unis d'Amérique, octobre 1995.

« An adaptive memory procedure for vehicle routing problems », *Metaheuristic International Conference*, Breckenridge, États-Unis d'Amérique, juillet 1995.

« Diversification, intensification and parallelization in local searches for vehicle routing problems », *Optimization Days*, Montréal, Canada, mai 1995.

« Comparison of efficient heuristic methods for the QAP », *Optimization Days*, Montréal, Canada, mai 1995.

« Decomposition of large vehicle routing problems », *European Doctoral Program in Quantitative Methods in Management*, Londres, Royaume-Uni, avril 1993.

« Parallel iterative search methods for the vehicle routing problem », Séminaire donné au *Politecnico di Milano*, Italie, juillet 1992.

« Adaptation of taboo search techniques to a real-life vehicle routing problem », *Rencontres franco-suisses de recherche opérationnelle*, Paris, France, octobre 1991.

« Robust taboo search for the quadratic assignment problem », *Viewpoints on Optimization*, Grimentz, Suisse, septembre 1990.

« Parallel taboo search techniques and applications », *European Doctoral Program in Quantitative Methods in Management*, Bruxelles, Belgique, avril 1990.

« Parallel taboo search techniques for the flow shop and job shop scheduling problems », *4th Advanced Research Institute on Discrete Applied Mathematics*, Rutgers University, États-Unis d'Amérique, mai 1989.

« Some efficient heuristic method for the flow shop sequencing problem », *European Doctoral Program in Quantitative Methods in Management*, Bruxelles, Belgique, avril 1989.

PARTICIPATION À DES COMITÉS DE LECTURE, JURY DE THÈSE.

Éditeur associé de *Journal of Heuristics*.

Co-éditeur, avec F. Glover, M. Laguna, et D. de Werra de *Tabu Search* (*Annals of Operations Research 41*).

Arbitre régulièrement des articles pour des revues d'envergure internationale comme *Annals of Operations Research, Computers and Operations Research, European Journal of Operational Research, IEEE, INFORMS Journal on Computing, Journal of Heuristics, Management Science, Operations Research, Studies in Locational Analysis, Transportation Science, …*

Membre du jury de la thèse de doctorat de **E. Angel**, *La rugosité des paysages : une théorie pour la difficulté des problèmes d'optimisation combinatoire relativement aux méta-heuristiques*, Université de Paris XI- U. F. R. des Sciences d'Orsay, France, 1998.

Membre du jury de la thèse de doctorat de **C. Rego**, *Local Search and Neighbourhood Structures for Vehicle Routing Problems : Sequential and Parallel Algorithms*, Université de Versailles, France, 1996.

Membre du jury de la « Tesi di Laurea » de **L. Mazzone** et **L. Romanelli**, *Algoritmi ispirati alle colonie di formiche applicati al problema dell'assegnamento quadratico*, Politecnico di Milano, Italie, 1996.

Annexe II : Articles et rapports techniques

P. Badeau, F. Guertin, M. Gendreau, J.-Y. Potvin, É. D. Taillard, « A parallel tabu search heuristic for the vehicle routing problem with time windows », *Transportation Research–C 5*, 1997, 109-122.

L. M. Gambardella, É. D. Taillard, M. Dorigo, « Ant colonies for the quadratic assignment problem », Rapport technique IDSIA-4-97, IDSIA, Lugano, 1997.

M. Gendreau, F. Guertin, J.-Y. Potvin, É. D. Taillard, « Tabu search for real-time vehicle routing and dispatching », Publication *CRT-96-47*, Centre de recherche sur les transports, Université de Montréal, 1996.

B. L. Golden, G. Laporte, É. D. Taillard, « An adaptive memory heuristic for a class of vehicle routing problems with minmax objective », *Computers and OR 24*, 1997, 445-452.

Y. Rochat, É. D. Taillard, « Probabilistic diversification and intensification in local search for vehicle routing », *Journal of Heuristics 1*, 1995, 147-167.

É. D. Taillard, « Robust taboo search for the quadratic assignment problem », *Parallel Computing 17*, 1991, 443-455.

Annexe II : Articles et rapports techniques (suite)

É. D. Taillard, « Parallel iterative search methods for vehicle routing problems », *Networks 23*, 1993, 661-673.

É. D. Taillard, « Comparison of iterative searches for the quadratic assignment problem », *Location science 3*, 1995, 87-105.

É. D. Taillard, G. Laporte, M. Gendreau, « Vehicle routing with multiple use of vehicles », *Journal of the Operational Research Society 47,* 1996, 1065-1070.

É. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, « A tabu search heuristic for the vehicle routing problem with soft time windows », *Transportation Science 31*, 1997, 170-186.

É. D. Taillard, L. M. Gambardella, « Adaptive memories for the quadratic assignment problem », Rapport technique IDSIA-97-97, IDSIA, Lugano, 1997.

É. D. Taillard, « A heuristic column generation method for the heterogeneous VRP », Publication *CRT-96-03*, Centre de recherche sur les transports, Université de Montréal, 1996. À paraître dans *RAIRO-OR*.

É. D. Taillard, « Heuristic methods for large centroid clustering problems », Rapport technique IDSIA-96-96, IDSIA, Lugano, 1996.