RÉPUBLIQUE ALGERIÉNNE DÉMOCRATIQUE ET POPULAIRE
MINISTÉRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université des sciences et de la technologie Houari Boumediene – USTHB
Faculté d'électronique & informatique
Département informatique

THÈSE

présentée pour obtenir

LE GRADE DE DOCTEUR D'ÉTAT EN SCIENCES DE L'USTHB

Spécialité : INFORMATIQUE

Par

Malek RAHOUAL

Sujet : **Contribution à l'optimisation combinatoire mono et multiobjectif par des méthodes coopératives : application aux problèmes d'ordonnancement.**

Soutenue le 17 décembre 2007

devant le jury composé de :

AHMED NACER Mohamed	Professeur à l'USTHB	*Co-directeur*
BADACHE Nadjib	Professeur à l'USTHB	*Président*
BERRACHEDI Abdelhafid	Professeur à l'USTHB	*Examinateur*
KONIG Jean-Claude	Professeur à l'université de Montpellier	*Directeur de thèse*
SAHNOUN Zaidi	Professeur à l'université de Constantine	*Examinateur*
TALBI El-Ghazali	Professeur à l'université de Lille	*Examinateur*

Résumé

Cette thèse décrit des travaux de recherche dans le domaine de l'optimisation combinatoire. Un problème combinatoire est d'habitude caractérisé par un ensemble de variables de décision discrètes et par une (respectivement plusieurs) fonction(s) objectif(s) qui associe(nt) à chaque solution (respectivement un ensemble de solutions) son coût (respectivement son vecteur coût), ou son intérêt. Il peut exister, par conséquent, un nombre fini, mais extrêmement grand, de solutions. Ainsi, dans un cadre monoobjectif, la résolution d'un problème d'optimisation combinatoire consiste à déterminer une solution qui optimise une fonction objectif donnée. Dans le cadre multiobjectif, la résolution d'un problème consiste à trouver un ensemble de solutions optimisant simultanément les différents objectifs.

Dans ce contexte, la classe de problèmes NP-difficiles rassemble des problèmes pour lesquels il n'existe pas d'algorithme qui fournit la (les) solution(s) optimale(s) en un temps polynomial en la taille de l'instance, sauf si P=NP. Face à cette situation, une question naturelle s'impose : quels plans d'attaque adopter pour résoudre un problème d'optimisation combinatoire, c'est-à-dire comment déterminer la (les) meilleure(s) solution(s) dans un espace de recherche vaste et ceci dans un temps *raisonnable* ?
Puisque le phénomène d'explosion combinatoire nous interdit l'exploration complète de l'espace de recherche dans un temps raisonnable, des méthodes ont été conçues pour restreindre la portion de l'espace explorée. Ces méthodes, dites *heuristiques*, s'appuient sur des propriétés spécifiques du problème à résoudre. Ces propriétés sont connues ou sont acquises par l'heuristique au cours de l'exploration de l'espace de recherche. Autrement dit, ce sont des recherches guidées par des "astuces" qui *dépendent* du problème traité. Ces heuristiques utilisent les particularités du problème traité afin d'engendrer, de manière partiellement stochastique ou non, de bonnes solutions. Ces approches sont appelées *heuristiques constructives*. Parmi ces approches, les algorithmes gloutons sont bien connus. Ils consistent à créer une (des) solution(s) pas à pas, en exploitant les particularités du problème. Pour le *problème du voyageur de commerce*, par exemple, un algorithme glouton basique serait de visiter, à chaque itération, la ville non visitée la plus proche.

Les *métaheuristiques*, telles que les *algorithmes génétiques*, le *recuit simulé*, les algorithmes de *colonies de fourmis*, la *méthode tabou* …, sont des méthodes qui ne sont pas spécifiques à un problème particulier ; elles n'ont aucune connaissance *a priori* du problème à résoudre. Le fonctionnement des métaheuristiques, centre d'intérêt de notre travail, est plutôt simple : d'une manière générale, ces méthodes s'appuient sur une ou plusieurs solutions, de qualités (coûts) quelconques, qu'elles tentent d'améliorer, pas à pas, au fil des itérations, en les modifiant partiellement.
Ces algorithmes intègrent des principes de résolution de problèmes combinatoires très généraux ; de ce fait, ils s'adaptent aisément à la résolution de nombreux problèmes. Cependant, la puissance d'une métaheuristique est d'abord liée à son aptitude à intégrer des connaissances spécifiques du problème afin d'améliorer ses performances. La connaissance du problème la plus fondamentale réside dans le codage du problème et dans le choix de la fonction de voisinage. Les métaheuristiques tentent également d'améliorer leur efficacité en incorporant des connaissances supplémentaires dans leurs opérateurs. Plus les opérateurs d'une méthode utilisent des connaissances spécifiques, plus la méthode dispose de moyens potentiels pour conduire efficacement la recherche. En contrepartie, l'intégration de ces connaissances spécifiques (en supposant que ces connaissances soient disponibles) nécessite un effort pour spécialiser ou adapter la méthode.

Depuis quelques années, on constate que les métaheuristiques les plus performantes sont des méthodes hybrides, c'est-à-dire des méthodes combinant deux ou plusieurs métaheuristiques. A l'origine de l'hybridation des métaheuristiques se trouvent des constatations selon lesquelles certaines méthodes auraient tendance à trop *explorer* l'espace de recherche (*diversification*), alors que d'autres, au contraire, auraient plutôt tendance à faire plus d'*exploitation* des solutions déjà rencontrées (*intensification*). Ainsi, la conjugaison, dans une méthode hybride, de métaheuristiques aux comportements complémentaires, fournirait l'équilibre recherché entre diversification et intensification, donnant une métaheuristique performante. D'autres approches de *coopération/hybridation*, entre métaheuristiques, heuristiques spécifiques ou approches exactes, sont apparues dernièrement. Une grande partie de ces travaux a comme objectif de proposer des coopérations non seulement performantes mais aussi réutilisables et permettant de dissocier, autant que possible, la représentation du problème de sa résolution. Dans cette perspective, nous nous sommes intéressés à une méthode de résolution exacte : *la programmation par contraintes*. Cette dernière est une technique générale de résolution de problèmes d'optimisation combinatoire. Elle se situe au carrefour de l'intelligence artificielle et de la recherche opérationnelle. L'efficacité de la programmation par contraintes tient au fait qu'elle permet de dissocier la représentation du problème (définition des contraintes et des objectifs) de sa résolution, réalisée par le système. Notre but est d'exploiter sa coopération avec les métaheuristiques.

Une approche complémentaire, pour obtenir des temps de calcul raisonnables, est l'emploi de calculateurs puissants, c'est-à-dire intégrant plusieurs processeurs. Ainsi, pour gagner en efficacité, il convient d'intégrer à l'optimisation combinatoire des techniques de parallélisme pour concevoir des méthodes d'optimisation parallèles plus performantes. En effet, l'utilisation du parallélisme a permis d'obtenir des résultats satisfaisants, sur des instances de grandes tailles, pour des problèmes d'optimisation tels que le problème du voyageur de commerce, le problème de tournées de véhicules, le problème de *job shop*, le problème de l'affectation quadratique, ou le problème de partitionnement de graphes, ….

Dans cette thèse, nous avons abordé le domaine de la coopération entre des méthodes d'optimisation, dans le cadre de la résolution de problèmes NP-difficiles de l'optimisation combinatoires *mono* et *multiobjectifs*. Nous nous sommes particulièrement intéressés aux métaheuristiques séquentielles et hybrides, ainsi qu'à leur parallélisation.

Dans le cadre *monoobjectif*, nous avons proposé différentes métaheuristiques hybrides, ainsi que des schémas de coopération génériques pour la résolution de différents problèmes classiques de l'optimisation combinatoire : un problème d'ordonnancement de tâches sur une architecture parallèle, le problème de couverture d'ensembles et un problème de bioinformatique, le repliement de protéines.

Le problème d'ordonnancement de tâches sur une architecture parallèle est fondamental pour la conception d'algorithmes parallèles. Nous avons considéré un problème d'ordonnancement statique formé d'un ensemble de tâches (sans duplication et sans préemption) à ordonnancer sur une architecture multiprocesseur à mémoire distribuée, avec comme fonction de coût la minimisation de la date de terminaison de la dernière tâche (*makespan*).
Nous avons proposé deux algorithmes de recherche tabou, ainsi qu'une version parallèle de cette approche. Le premier algorithme est une méthode *Tabou* dans sa forme classique. Le second est une amélioration du premier, dans la mesure où il intègre des étapes

d'intensification et de diversification. Le troisième algorithme est un algorithme parallèle d'une portée générale. En effet, d'une part, il est indépendant de l'architecture sur laquelle il s'exécute, et d'autre part, aucune hypothèse n'est faite sur la structure de l'application. Il s'agit d'un algorithme synchrone. Il est obtenu en particularisant le rôle de l'un des processus (le maître), qui est chargé de distribuer des solutions initiales et des stratégies entre les différents processus esclaves. Chacun de ces processus esclaves applique alors un algorithme de recherche tabou sur la solution initiale qui lui a été attribuée par le maître, et selon la stratégie choisie par ce dernier également. Les solutions trouvées par les esclaves sont ensuite envoyées au processus maître qui sélectionne la meilleure d'entre elles.
D'après les tests effectués sur les différents algorithmes développés, nous avons remarqué que les résultats obtenus dépendaient de la nature du graphe de précédence (durées d'exécution des tâches, coûts de communication entre paires de tâches, densité du graphe). En effet, lorsque les coûts de communication sont faibles, les tâches se répartissent assez bien sur les différents processeurs. Dans le cas contraire, elles ont tendance à s'exécuter sur un nombre restreint de processeurs. Par ailleurs, nous avons montré que le problème est NP-complet même pour des cas particuliers qui nous intéresse. Nous avons également montré que le problème est dur au regard de l'approximation dans le sens qu'il n'admet pas de schéma d'approximation polynomial à moins que P=NP. Ainsi, nous avons adopté une approche plus pratique à ce problème.

Au travers du problème de couverture d'ensembles, deux algorithmes parallèles à base de colonies de fourmis et deux hybridations avec la recherche locale ont été proposés. Dans un premier temps, nous avons proposé un algorithme de colonies de fourmis simple. Cet algorithme a ensuite été hybridé avec une recherche locale. Les deux types d'hybridations qui ont été mises en œuvre sont l'hybridation *HRH* (*High-Level Relay Hybrid*) et l'hybridation *HCH (High Co-evolutionary Hybrid)*. Dans les hybridations de type *HRH*, chaque métaheuristique qui intervient dans ce processus d'hybridation reste inchangée et autonome. Les méthodes *HRH*-hybridées sont exécutées en séquence. Au contraire, dans les hybridations *HCH*, les algorithmes hybridés, dont la structure interne n'est pas modifiée, sont exécutés simultanément et coopèrent pour résoudre le problème. Les tests effectués ont montré que la deuxième hybridation était meilleure que la première. Les résultats obtenus ont ensuite été améliorés en utilisant différentes implémentations parallèles de l'algorithme hybride le plus performant. Ces implémentations parallèles ont permis de réduire le temps de recherche ; cependant nous avons remarqué que l'efficacité de la parallélisation dépendait de la taille du problème traité. Pour le problème de couverture d'ensembles, la valeur de l'accélération (*speedup)* n'était pas uniformément distribuée par rapport aux instances traitées. Cette valeur augmentait proportionnellement avec la taille de l'instance traitée. L'augmentation de l'accélération s'explique par le fait que le temps de communication, entre le processus maître et les processus esclaves, est très élevé par rapport au temps de traitement sur chacun des processus esclaves pour les instances de petite taille. Par ailleurs, pour les instances de grande taille, les temps de communication sont faibles.

Dans le cadre de la coopération entre une méthode exacte et une métaheuristique, nous avons proposé un certain nombre d'algorithmes. Ces derniers font coopérer la programmation par contraintes avec les colonies de fourmis. Ces algorithmes ont été développés sous l'environnement de la programmation par contraintes *Mozart-Oz* et appliqués au problème de repliement de protéines.
Deux modèles à base de la programmation par contraintes ont été proposés. Le premier, simple et de base, est insuffisant, même hybridé avec une recherche locale basée sur le voisinage *Pull-Move*. Un autre modèle, plus élaboré, a été proposé. Ce modèle introduit de

nouvelles contraintes basées sur la notion de "cadre". Il ne donne des résultats satisfaisants que pour les protéines de taille inférieure à 50. Pour pallier à ce manque, nous avons proposé un schéma de coopération entre la programmation par contraintes et les colonies de fourmis. Cette coopération permet une meilleure recherche de la solution, en réduisant l'espace de recherche exploré par la programmation par contraintes de deux manières. D'une part, il est possible d'exploiter les renseignements quant à une solution déjà calculée. Ces renseignements sont exprimés par des contraintes additionnelles : après qu'une solution ait été trouvée, la contrainte supplémentaire qu'une prochaine solution doit être meilleure est prise en considération. Avec cette contrainte supplémentaire, l'arbre de recherche peut devenir considérablement plus petit. D'autre part, les valeurs de taux de phéromone calculés par les fourmis, sont utilisées pour explorer l'arbre de recherche. Il s'agit en fait de choisir l'ordre des directions à explorer pour chaque monomère, et ainsi commencer par explorer les directions les plus prometteuses (c'est-à-dire la branche sur laquelle la valeur de la phéromone est la plus importante). Les résultats de cette coopération entre une métaheuristique et la programmation par contraintes sont satisfaisants. Le temps de calcul de la programmation par contraintes est d'autant plus réduit que la recherche par colonies de fourmis est de qualité.
Enfin, cette étude a permis de constater que ces algorithmes génériques peuvent être appliqués à d'autres problèmes d'optimisation, moyennant, éventuellement, des modifications spécifiques à la problématique. En effet, la programmation par contraintes permet l'ajout de contraintes au problème initial, sans que cela n'altère le code déjà développé, pour la coopération avec les colonies de fourmis.

Dans le cas *multiobjectif*, nous avons proposé des algorithmes génétiques ayant pour objectif d'approcher la frontière de Pareto. A cet effet, plusieurs mécanismes ont été introduits dans notre algorithme : l'élitisme, l'hybridation, le parallélisme, la diversification génotypique, la diversification phénotypique, etc. Nos algorithmes ont été testés sur les problèmes du *flow-shop* ainsi que le problème de tournées de véhicules avec des fenêtres horaires.

Le problème du *flow-shop* à permutation bi-objectif a été traité, avec comme objectif la minimisation du temps de terminaison (*makespan*) et la somme des retards de toutes les tâches. Nous avons utilisé pour sa résolution des algorithmes génétiques. Les tests effectués sur cinq stratégies de sélection montrent que les stratégies Pareto (NSGA, NDS, WAR) sont celles qui fournissent les meilleurs résultats. Ces stratégies partagent en commun le fait de s'appuyer sur la notion de dominance, afin de classer entres elles des solutions de la population.
D'une manière générale, les algorithmes génétiques sont souvent critiqués pour leur lenteur. L'hybridation de la méthode avec la recherche locale offre alors une solution à ce problème. L'algorithme génétique proposé est lancé en premier, fournissant une première approximation de la frontière de Pareto, la recherche locale est ensuite lancée avec pour entrée chacune des solutions de l'ensemble Pareto fourni par l'algorithme génétique. L'effet de cette hybridation, du type HRH, n'est remarquable que pour des problèmes de grande taille.
Pour réduire les temps de réponse et augmenter la taille de la population, un modèle d'algorithme génétique parallèle a été proposé. Il se base sur une approche distribuée. Dans ce cas, plusieurs algorithmes génétiques sont lancés, chacun ayant pour entrée une sous-population initiale d'individus. A des périodes de temps déterminées, une migration circulaire est effectuée entre les différents processus de recherche.
Les algorithmes génétiques sont aussi connus pour leur sensibilité quant au choix de la population initiale. Nous avons alors proposé une adaptation probabiliste de l'heuristique *NEH*. Ainsi, la moitié des individus de cette population ont été construits de façon à présenter

de bons temps de terminaison (*makespan*), l'autre moitié étant construits afin d'offrir de bons temps de retard.
L'absence de benchmarks pour les problèmes multiobjectifs est souvent une difficulté à laquelle doit faire face le concepteur d'algorithmes d'optimisation. Nous avons proposé dans cette thèse un générateur d'instances pour le problème *flow-shop* bi-objectif. Ce générateur se présente comme une adaptation de l'algorithme de *Taillard,* souvent référencé dans la littérature. Nous estimons que ces instances fourniront une bonne base de comparaison pour les travaux futurs. Une batterie de tests ont été effectués, démontrant une certaine habilité de nos algorithmes à trouver des solutions de faible compromis (solutions extrêmes).

Enfin, nous avons étudié le problème de tournées de véhicules avec des fenêtres horaires. Notre contribution a porté sur l'adaptation d'algorithmes génétiques hybrides au VRPTW bi-objectif (distance parcourue, nombre de véhicules), en utilisant l'approche Pareto. Nous avons enrichi l'algorithme génétique de base par l'introduction des éléments suivants :

- une heuristique pour la génération de la population initiale,
- des opérateurs de croisement (*RBX* et *SBX*),
- des opérateurs de mutation (*swap* (*reinsert*) et *One Level Exchange*),
- des approches de sélection (*NSGA* et *ELITISME*),
- des techniques de partage *(sharing)* pour la diversification,
- un algorithme hybride avec la recherche locale.

Les algorithmes développés ont été testés sur les instances de Solomon. Les solutions extrêmes de la courbe de Pareto ont été comparées aux meilleurs résultats monoobjectifs de la littérature. Il en ressort que nos solutions sont de bonne qualité pour les instances de la classe C1 et mitigées pour les autres classes du fait de leur nature et de leur complexité. Cependant, nos courbes de Pareto offrent aux décideurs un éventail de solutions, leur permettant ainsi de faire un choix de compromis.
Par ailleurs, les résultats de complexité obtenus indiquent clairement que le problème considéré est fortement intraitable pour l'approximation, excluant jusqu'à l'existence d'un rapport d'approximation inférieur à N^{α}, pour tout $\alpha \prec 1$, à moins que P=NP, où N est le nombre de sites. Ainsi, le mieux que l'on puisse espérer à cet égard, est d'obtenir un rapport d'approximation qui soit fonction des fenêtres de temps $[\alpha(i),ß(i)]$, sans que ce rapport ne soit, là encore, inférieur à $(\max_i ß(i)/\max_i \alpha(i))$ (à moins que P=NP).
Par ailleurs nous avons indiqué comment les puissantes méthodes de coupes de la programmation linéaires, qui ont fait leurs preuves dans le traitement du TSP, peuvent être appliquées dans un contexte de fenêtres de temps.

Parmi les extensions possibles de ces travaux, aussi bien dans le cas monoobjectif que dans le cas multiobjectif, on peut citer :

- l'adaptation des algorithmes génériques développés entre la programmation par contraintes et la métaheuristique des colonies de fourmis à d'autres problèmes d'optimisation ;
- la prise en charge et le développement d'outils de résolution de problèmes d'optimisation multiobjectif comprenant plus de deux objectifs ;
- Dans le cadre de l'hybridation de métaheuristiques avec la recherche locale, il serait intéressant de développer des techniques d'exploration de grands voisinages de tailles exponentielles en temps polynomial. On peut notamment penser à la programmation dynamique, qui a été utilisée avec succès pour le problème de voyageur de commerce, pour explorer des voisinages de grande taille ;

- Il serait intéressant d'explorer d'autres pistes de coopération entre des méthodes exactes et des métaheuristiques ;
- La proposition d'autres techniques de parallélisation des méthodes hybrides est possible ;
- Le développement de métaheuristiques adaptatives (méthodes mimétiques, opérateurs de mutation spécialisés et adaptatifs, etc.) est un axe à explorer pour la résolution de problèmes d'optimisation combinatoire d'une manière plus efficace ;
- Enfin, il est sans doute possible de mener une analyse plus rigoureuse des résultats obtenus dans le cadre de l'optimisation multiobjectif, en utilisant les différentes métriques de la littérature et en utilisant des tests statistiques.

Nous travaillons, actuellement, sur l'amélioration des approches de coopération proposées entre la programmation par contraintes et les colonies de fourmis ainsi que sur l'exploration d'autres types de coopération.