NNE DEMOCRATIQUE ET POPULAIRE
LôENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE DOCTORAL DE LôEST

MEMOIRE

Présenté à l'université de OUM EL BOUAGHUI

Faculté des sciences de l'ingénieur - Département d'informatique

Pour l'obtention du diplôme

Magister en informatique

OPTION
INTELLIGENCE ARTIFICIELLE

Par

DEFFAS ZINEB

Thème

Métaheuristiques parallèles à solution unique pour la résolution du problème du Q3AP sur grille de calcul

Soutenue le : 29/04/2010 Devant le jury composé de :

Dr A. Chaoui	Président	Maître de conférences (UMC)
Dr M. K. Kholladi	Rapporteur	Maître de conférences (UMC)
Dr S. Chikhi	Examinateur	Maître de conférences (UMC)
Dr D. E. Saïdouni	Examinateur	Maître de conférences (UMC)

Remerciements

Je remercie Dieu tout puissant clément et miséricordieux de m'avoir soigné et aidé.

Je tiens, avant tout, à exprimer ma profonde gratitude au Dr. Mohamed-Khireddine Kholladi Maître de Conférences à l'université de Constantine, qui a assuré la direction de ce travail.

Je remercie les membres de jury qui ont accepté de juger ce travail et d'y apporter leur caution :

Dr. A. Chaoui, maître de conférences à l'université de Constantine, qui me fait le grand honneur d'accepter la présidence du jury.

Dr. S. Chikhi, maître de conférences à l'université de Constantine, pour l'honneur qu'il me fait en acceptant de participer à ce jury.

Dr. D. E. Saidouni, maître de conférences à l'université de Constantine, pour l'honneur qu'il me fait en acceptant également de participer à ce jury.

J'adresse mes vifs remerciements à tous les membres d'équipe de Mr kholladi, par leur encouragement et leur aide, surtout Mr C. Mezieud et Mr A. S. Layeb.

Dédicaces

Je remercie Dieu pour m'avoir donné la force d'accomplir ce travail pour aller plus loin.

Je dédie ce travail à mes parents, ma mère pour ses encouragements et ses prières tout au long de mes études, mon père pour tout ce qu'il avait fait pour avoir ce résultat.

Je le dédie à mes frères et sœurs, et je les remercie pour leurs encouragements et leurs aides ainsi que toute ma grande famille.

A mes collègues de travail de l'Algérie télécoms de ferdjioua.

A tous mes amis sans citer les noms.

A mes collègues de l'école doctorale promotion 2006.

A tous ceux qui aiment Zineb et ceux qui Zineb aime.

Résumé

Dans le domaine de lôoptimisation combinatoire, la résolution du problème d'affectation quadratique à trois dimensions (Q3AP) fait lôobjet de recherche intensive. Ce dernier est une extension à trois dimensions du problème bien connu d'affectation quadratique (QAP).

Deux grandes familles de méthodes de résolution ont été proposées. La première englobe les méthodes, dites exactes, qui ont l'avantage de garantir l'optimalité des solutions si elles existent mais souffrent, cependant, de l'explosion combinatoire. La seconde famille de méthodes, dites approchées, permet de réduire cette explosion sans toutefois garantir l'optimalité. C'est dans le cadre de cette famille que se place notre travaille où se trouve les métaheuristiques.

Nous proposons deux métaheuristiques recuit simulé et recherche Tabou pour résoudre Q3AP, Les deux méthodes sont relativement faciles à mettre en î uvre, mais elles nécessitent un certain savoir-faire pour régler les différents paramètres. Nous introduirons la notion de parallélisme afin d'ajuster leur convergence et, par là même, accélérer l'obtention d'une bonne solution.

Mots clés : Optimisation combinatoire, problème dôaffectation quadratique à trois dimensions, explosion combinatoire, métaheuristique, Recuit Simulé, Recherche Tabou.

Abstract

In the domain of the combinatorial optimization, the resolution of the Quadratic 3-dimensional Assignment Problem (Q3AP) is the subject of intensive research. This is an extension to three dimensions of well-known of quadratic assignment problem (QAP).

Two large families of solution methods have been proposed. The first includes methods, called exact, which have the advantage of assure the optimality of solutions if they exist, but suffer, however, the combinatorial explosion. The second family of methods, called approximate, reduces the explosion without assure optimality. Is through this family that is our work place where found metaheuristics.

We propose two metaheuristics Simulated Annealing and Tabu Search to solve Q3AP, Both methods are relatively easy to implement, but they require a certain expertise to adjust the various parameters. We introduce the concept of parallelism to adjust convergence and thereby accelerate the obtaining of a good solution.

Key words: Combinatorial optimization, quadratic 3-dimensional assignment problem, combinatorial explosion, metaheuristic, Simulated Annealing, Tabu Search.

ملخص

في مجال التحسين التوافقي ، حل مسألة التخصيص التربيعية بثلاثة أبعاد (Q3AP) هي موضوع البحث المكثف. هذا الأخير هو امتداد بثلاثة أبعاد للمسألة المعروفة بالتخصيص التربيعية (QAP).

هناك مجموعتان كبيرتان من طرق الحل. المجموعة الأولة تضم الطرق الدقيقة ، التي تتميز بضمان الحل الأمثل إن وجد ، ولكنها تعاني من مشكل الانفجار التوافقي. المجموعة الثانية تسمى الطرق التقريبية و التي تعالج هذه المسألة دون ضمان الحل الأمثل. ضمن هذه المجموعة يندرج عملنا أين نجد الطرق الاستكشافية (العشوائية) .

في هذا العمل نقترح طريقتين، التردد التمثيلي و البحث المحظور من أجل حل Q3AP، كلتا الطريقتين سهلتين نسبيا للتنفيذ، ولكنهما تتطلبان بعض المعارف لتحديد مختلف العناصر. كما أننا ندخل مفهوم التوازي من أجل ضبط التقارب، وكذلك من أجل التعجيل في الوصول إلى حل جيد.

الكلمات المفتاحية: التحسين التوافقي، مسألة التخصيص التربيعية بثلاثة أبعاد ، الانفجار التوافقي، الطرق الاستكشافية ، التردد التمثيلي ، البحث المحظور.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

méthodes de recherche pour arriver aux meilleurs résultats en qualité et en temps de traitement a attiré l'attention depuis l'apparition d'un type de problèmes classés difficiles. De plus en plus, le besoin d'améliorer ces méthodes de recherche ou de trouver de nouvelles méthodes est devenu indispensable. Dans le domaine de recherche en intelligence artificielle, lôoptimisation combinatoire occupe une place importante. Les problèmes quôelle traite sont souvent difficiles et demandent un temps important de résolution qui croît exponentiellement avec la taille des instances du problème.

Le problème dôaffectation quadratique est lôun des problèmes dôoptimisation combinatoire qui a prouvé dans la pratique quôil est le plus difficile à résoudre jusquôà lôoptimum, il convient de trouver le placement optimal de " n " usines communiquant entre elles sur " n " sites prédéterminés de façon à minimiser un coût quadratique dépendant à la fois des distances inter-sites et des flux inter-usines. Dans ce travail nous sommes intéressés au problème dôaffectation quadratique à trois dimensions (Q3AP) qui est une extension du problème dôaffectation quadratique (QAP). Le Q3AP a été introduit récemment pour modéliser un protocole de tolérance aux pannes émergeant dans le domaine des télécommunications. Ce protocole, appelé Hybride ARQ (*Hybrid Automatic Repeat reQuest* ou *HARQ*). HARQ consiste á retransmettre le signal original en cas d'erreur détectée, en utilisant á chaque retransmission un codage différent. Dans l'HARQ, l'idée est d'utiliser des combinaisons (symbole QAM ï code) binaires différents à chaque retransmission de façon à créer une diversité et à minimiser le BER (Bit Error Rate).

Plusieurs méthodes de recherche ont été implémentées pour résoudre les problèmes dôoptimisation combinatoire. Ces méthodes se regroupent en deux grandes familles : La première englobe les méthodes, dites exactes, basées sur un parcours implicite de tout lôespace de recherche pour donner une réponse exacte, elles garantirent l'optimalité des solutions si elles existent mais souffrent, cependant, de l'explosion combinatoire. La seconde famille de méthodes, dites approchées, permet de réduire cette explosion sans toutefois garantir l'optimalité, elles ne parcourent pas tout lôespace de recherche mais elles utilisent des techniques pour le parcourir. Parmi ces méthodes on trouve les heuristiques et les métaheuristiques. Ces dernières données sous forme de concepts généraux qui permettent de résoudre des problèmes indépendemment de leur nature.

Les métaheuristiques classées généralement en deux catégories: celles à population de solutions comme les algorithmes évolutionnaires, les systèmes de fourmis et la recherche par dispersion et les autres à base de solution unique qui regroupent essentiellement les GRASPs, les méthodes de recuit simulé et de recherche tabou. Bien que les métaheuristiques ont connu un grand succès en résolution des problèmes difficiles, mais elles soufrent avec les problèmes de très grande taille. De plus, il existe des cas où elles n'arrivent pas à trouver les bons résultats. Le besoin dôaccélérer le traitement n'est pas dû seulement à la grande taille des instances mais aussi à l'utilisation d'une configuration difficile pour les métaheuristiques comme l'augmentation du nombre des itérations qui a pour but d'améliorer la qualité.

De ce qui précède, le traitement parallèle peut nous aider à améliorer les performances des métaheuristiques. Dans ce cadre, plusieurs stratégies de parallèlisation ont été proposées. Parmi elles, celles qui sont plus convenables avec un type de métaheuristiques ou avec une architecture spécifique de machines parallèles. Au contraire du concept classique du traitement parallèle, il existe un type de ces stratégies qui ne trouvent pas forcément le même résultat que l'implémentation séquentielle. Donc, ce type de stratégies peut améliorer les performances non seulement en temps mais aussi en qualité de la solution.

Notre travail consiste à étudier deux métaheuristiques « recuit simulé et recherche tabou » et les stratégies de parallèlisation des métaheuristiques, en premier lieu. Ensuite, essayer de construire un nouvelle schéma pour paralléliser chacune des deux méthodes étudie. Enfin, implémenter cette stratégie sur les deux méthodes et d'étudiées les performances de cette implémentation avec le Q3AP.

Dans cette thèse, le premier chapitre fait un état de lôart sur les problèmes dôoptimisation, tout en montrant la manière de définir un problème pareil, ses contraintes, et ses objectifs. Ainsi quelques exemples des problèmes dôoptimisation combinatoire, et nous allons parler aussi de leurs différentes méthodes de résolution existant. Dans le deuxième chapitre, une étude approfondie de métaheuristiques est décrite en présentant leurs caractéristiques et un panorama des stratégies pour le paralléliser. Le troisième chapitre présente une proposition de la résolution du problème dôaffectation quadratique en trois dimensions Q3AP par la méthode de recuit simulé et la méthode de recherche tabou. Ainsi, nous avons expliqué notre proposition de parallélisation des deux

randes lignes de ces métaheuristiques. Le dernier chapitre illustre les résultats obtenus au sein de la plate-forme ParadisEO qui est dédiée à la conception flexible et réalisable de métaheuristiques parallèles suivis par une comparaison entre les deux méthodes étudiées. Nous terminerons cette thèse par une conclusion générale permettant de mettre en évidence les perspectives envisagées pour des travaux de future.

Chapitre 1

Problème d'optimisation combinatoire

et

méthodes de résolution

Les ingénieurs se heurtent quotidiennement à des problèmes technologiques de complexité grandissante, qui surgissent dans des secteurs très divers, comme dans le traitement des images, la conception des systèmes, le design de réseaux de télécommunication la bio-informatique,é . Le problème à résoudre peut fréquemment être exprimé sous la forme générale dûun problème dôoptimisation, dans lequel on définit une fonction objective ou fonction de coût (voir plusieurs), que lôon cherche à optimiser (maximiser, minimiser) par rapport à tous les paramètres concernés.

La plupart des problèmes dôoptimisation combinatoire appartiennent à la classe des problèmes NP-difficiles, ce qui complique leur résolution de manière exacte et laisse place aux méthodes de résolution approchée par des heuristiques ou des métaheuristiques.

Tous ces concepts autour d'un problème d'optimisation combinatoire et le problème dôaffectation quadratique et son extension sont décrits dans ce qui suit tout en renforçant nos idées par des exemples et des méthodes utilisées pour les mettre en î uvre.

1.2. Définition

1.2.1. Un problème

Un problème du point de vue informatique (Computational Problem) veut dire comment faire relier un ensemble de données par un ensemble de résultats, où les données vont subir un ensemble d'opérations quôon appelle traitement. Donc il est conçu comme une relation entre les entrées (instances), et les sorties (solutions).

Pour quûne instance dûun problème soit compréhensible par un ordinateur numérique, elle doit être décrite comme une séquence finie de symboles dûun ensemble fini arbitraire appelé alphabet. De même, la solution dûne instance dûun tel problème est émise dans ce format **[1]**.

1.2.2. Un problème d'optimisation combinatoire

Un problème dôoptimisation combinatoire est toute situation dont on cherche à avoir une solution tout en respectant la présence d'un ensemble de contraintes. La solution consiste à combiner ces contraintes ensemble d'une manière à maximiser quelques uns et à minimiser les autres, ces contraintes ont une caractéristique primordiale, c'est que chaque contrainte influe sur les autres soit quand on minimise sa valeur ou on la maximise, dans un autre terme on dit que les contraintes sont conflictuelles.

e la *Figure 1.1* présente une situation de problème veut acheter une voiture moderne de meilleure qualité et en même temps à un prix raisonnable qui ne peut pas dépasser certaine limite. Si on maximise la première contrainte (une bonne voiture) on va avoir un prix maximale, dans le contraire on va aboutir à une mauvaise voiture mais avec un prix minimale dans les limites; on constate dans cet exemple que c'est difficile d'arranger ces deux contraintes dans nos besoins.

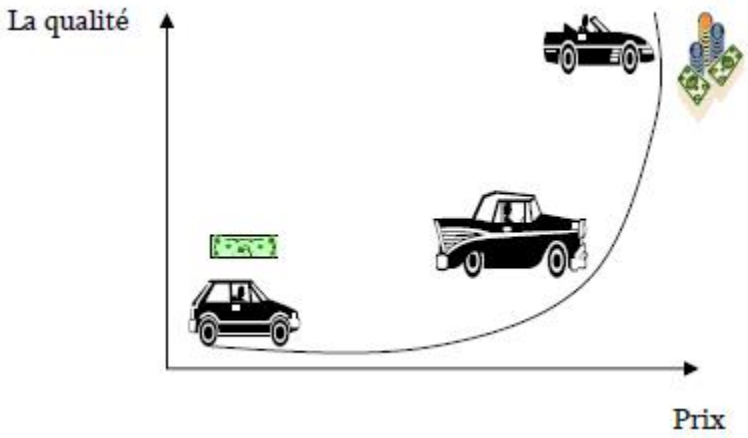

Figure 1.1 : Exemple d'un problème combinatoire [2].

Les problèmes d'optimisation combinatoire occupent actuellement une place de choix dans la communauté scientifique. Non pas qu'ils aient été un jour considérés comme secondaires mais l'évolution des techniques informatiques a permis de dynamiser les recherches dans ce domaine.

1.2.3. Caractéristiques d'un problème d'optimisation

Un problème d'optimisation combinatoire est défini par un espace d'état, une ou plusieurs fonction(s) objectif(s) et un ensemble de contraintes.

- L'espace d'état est défini par l'ensemble de domaines de définition des variables du problème.
- Les variables du problème peuvent être de nature diverse (réelle, entière, booléenne, etc.) et sont exprimées de données qualitatives ou quantitatives.
- Une fonction objectif représente le but à atteindre pour le décideur (minimisation de coût, de durée, d'erreur, ...). Elle définit un espace de solutions potentielles au problème.
- L'ensemble des contraintes définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes

ité et permettent en général de limiter l'espace de

La séparation entre les fonctions objectives et les contraintes peuvent paraître artificielles car nous pourrions considérer quôune contrainte est un objectif à atteindre. Mais elle se justifie de deux manières différentes. Dôune part, les contraintes sont appliquées sur lôespace de recherche alors que les objectifs définissent lôespace des solutions. Dôautre part, dans de nombreuses méthodes les contraintes et les objectifs sont traités par des procédures différentes.

- Une méthode d'optimisation cherche le point ou un ensemble de points de l'espace dôétat possible qui satisfait au mieux un ou plusieurs critères. Le *résultat est appelé valeur optimale (optimum).*

On peut dire quôun problème d'optimisation est défini comme la recherche du minimum ou du maximum (de l'optimum) d'une fonction donnée. On peut aussi trouver des problèmes d'optimisation pour lesquels les variables de la fonction à optimiser sont des contraintes à évoluer dans une certaine partie de l'espace de recherche **[3]**.

1.3. Exemples de problèmes d'optimisation combinatoires

Le monde réel offre un ensemble très divers de problèmes dôoptimisation combinatoire dans différent domaines.

1.3.1. Le problème de voyageur de commerce (PVC)

Parmi les problèmes les plus célèbres et connus dans le domaine dôoptimisation combinatoire le problème de Voyageur de Commerce (Traveling Salesman Problem, TSP). Il sôagit de minimiser la longueur de la tournée dôun représentant de commerce, qui doit visiter un certain nombre de villes " n ", avant de revenir dans la ville de départ. Le but du problème consiste à trouver le chemin le plus court pour parcourir un graphe de " n " nî uds en ne passant quôune seule fois par chaque nî ud (i.e. circuit hamiltonien) **[4]** *(cf.Figure 1.2)*.

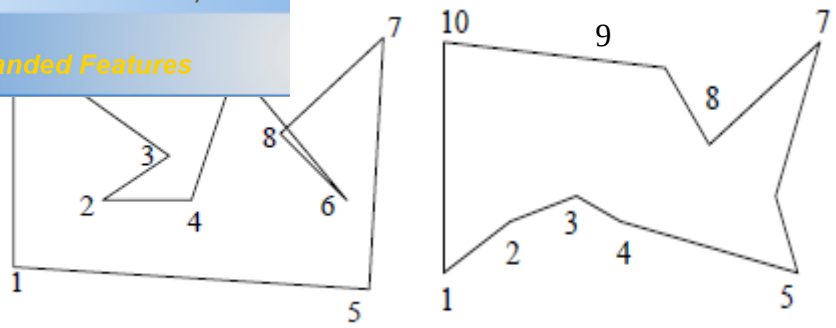

Figure 1.2 : Le tour d'un voyageur de commerce dans 10 villes.

1.3.2. Le problème de sac à dos (knapsack Problem)

Un randonneur doit décider de l'équipement qu'il veut emporter pour sa prochaine expédition. Pour des raisons évidentes, il veut limiter le poids total des objets emportés et s'est fixé une borne supérieure de b kg à ne pas dépasser.

Chacun des " n " objets qu'il peut emporter possède un poids " a_i ", i=1,é ,n ainsi qu'une utilité " c_i ", i=1,é ,n, cette dernière étant d'autant plus grande que le randonneur juge l'objet intéressant **[5]**.

1.3.3. Problème SAT (Problème de satisfaction)

Le problème de satisfiabilité (SAT) Le problème de satisfiabilité est l'un des problèmes de l'optimisation combinatoire. Elle consiste à trouver une affectation booléenne validant une formule en logique propositionnelle. Une instance de ce problème est définie par un ensemble de variables booléennes $X = \{x_1, \ldots, x_n\}$ et une formule booléenne $\emptyset : \{0,1\}^n \rightarrow \{0,1\}$.

Un littéral est une variable ou sa négation. Une affectation est une fonction $v : X \rightarrow \{0,1\}$. La formule $\emptyset$ est en forme normale conjonctive (CNF) si c'est une conjonction de clauses où une clause est une disjonction de littéraux **[6]**.

Il y a plusieurs variantes de ce problème comme le MAX-SAT, qui consiste à trouver une instanciation qui maximise le nombre des clauses satisfaites. Le MAX-W-SAT est une autre variante qui assigne des poids pour les clauses dans le but de maximiser la somme des poids des clauses satisfaites [7].

de graphe (graph coloring)

C'est un problème parmi les problèmes NP-Complet grâce à la difficulté trouvée lors de sa résolution, il consiste d'affecter un nombre défini de couleurs k aux sommets d'un graphe non orienté d'une manière à ce que les couleurs des nî uds adjacents soient différents. La coloration minimale utilise le petit nombre possible de couleur (couleur chromatique). La version décisive de coloration de graphe (k-coloring) demande quels sommets dans le graphe peuvent être colorés en utilisant un nombre Òk couleurs pour un k connu **[8]**.

En utilisant ce problème quelques applications sont résolues dans un temps raisonnable comme :

Les problèmes de « Time Tabling And Scheduling » fréquemment rejettent certaines tâches en parallèle à cause les dépendances entre computations. Planification des programmes avec le minimum hardware peuvent fréquemment être formalisée sous forme du problème de coloration de graphe.

Le problème dôallocation de registres: qui cherche à assigner des variables à un nombre limité de registres durant lôexécution de programme. Deux variables ne peuvent pas être assignées dans le même registre si elles sont « vive » en même temps. Lôassignement de variables en plus amène à lôexécution rapide comme moins de variables nécessitent dôêtre rapportées de la mémoire. Pour formaliser ce problème, on crée un graphe dont les nî uds représentent les variables et les arêtes représentent les conflits entre les variables.

Des diagrammes colorés pour lôassignement de libre conflit, et si le nombre de registres dépasse le nombre chromatique, un assignement de registre libre existe **[9]** *(cf.Figure 1.3).*

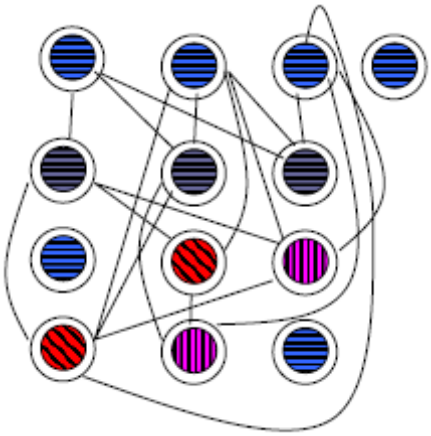

Figure 1.3 : Exemple de graphe coloré.

Dans le problème de 8-Queens, on a un tableau de jeu dôéchec régulier (8 par 8) et huit rênes qui doivent être placées dans le tableau dans un état dont aucune reine ne touche lôautre. Ce problème peut être naturellement généralisé, qui cède facilement le problème de N-queens. Beaucoup dôapproches classiques dôIA, qui sôappliquent à ce problème, travaillent dôune manière constructive ou incrémentale : un commence par placer une reine et après placer les " n " reines, un tente de placer la (nième +1) reine dans la meilleure position, c'est-à-dire une position dont la nouvelle reine ne peut toucher les autres **[10]** *(cf.Figure 1.4)*.

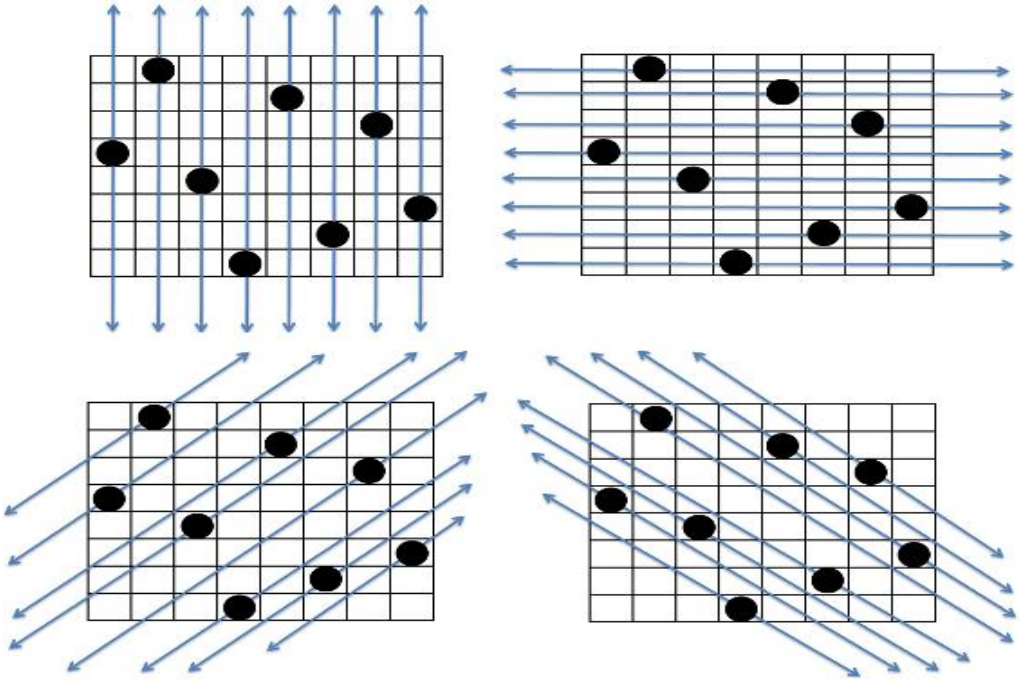

Figure 1.4 : L'illustration de la solution du problème de 8-Queens.

1.3.6. Problème d'Affectation quadratique

Le problème d'Affectation Quadratique est lôun des problèmes d'Optimisation Combinatoire les plus connus et ayant suscité un grand nombre de travaux, il est prouvé dans la pratique quôil est lôun des plus difficiles problèmes à résoudre jusquôà lôoptimum. Il a été introduit pour la première fois par Koopmans et Beckmann en 1957 afin de trouver une configuration optimale dôimplantation de " n " unités (usines, services hospitaliers...) communiquant entre elles sur " n " sites prédéterminés de façon à minimiser un coût quadratique dépendant à la fois des distances inter-sites et des flux inter-unités **[11]** *(cf.Figure 1.5)*.

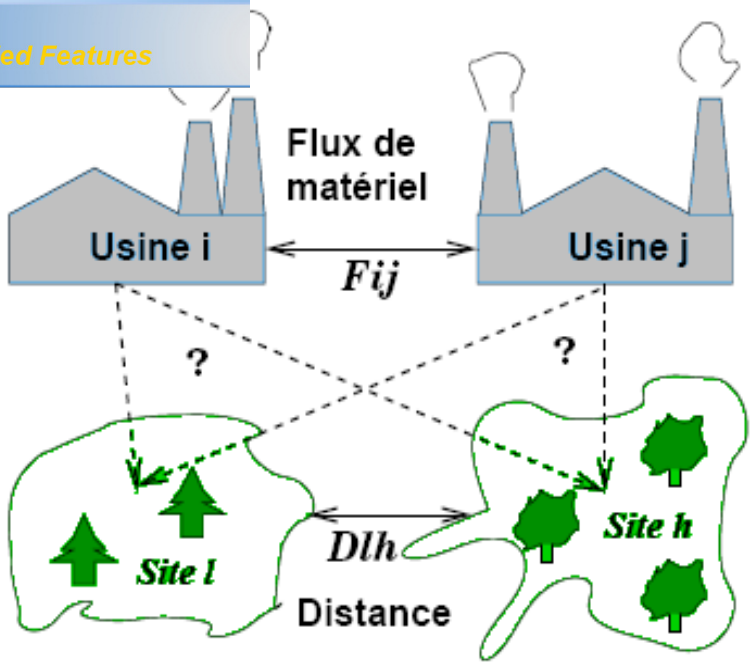

Figure 1.5 : Problème d'affectation Quadratique (QAP).

Pour prendre un exemple concret, cela correspond à une entreprise qui possède plusieurs usines de production qui *interagissent* entre elles. L'usine 2 a besoin du produit sorti de l'usine 1 pour pouvoir faire son travail. Ainsi, il va se créer un flux de marchandises entre ces usines, qui vont lui-même générer des coûts financiers pour la firme. Celle-ci a donc tout intérêt à bien répartir ses différentes usines sur les terrains qu'elle possède de manière à diminuer le prix du transport des marchandises entre les sites, et donc ses coûts de production. Plus le flux entre deux usines sera grand, plus la distance entre elles devra être petite.

1.3.7 Affectation quadratique en trois dimensions

Le Q3AP est une extension du QAP qui est une modélisation d'un problème stratégique qui vient d'une nouvelle application pratique en transmission de données sans fil (comme les réseaux de senseurs) : l'implantation d'un schéma hybride ARQ (Automatic Repeat reQuest) pour enrichir la diversité de transmissions multiples par paquet en optimisant le « mapping » des symboles de transmission aux données.

La solution du Q3AP peut réduire significativement le coût d'obtention d'une transmission fiable sur des canaux de communication sans fil avec distorsion du signal. Il s'écrit comme la minimisation d'une fonction quadratique sur le polytope d'affectation en 3 dimensions **[12]** *(cf.Figure 1.6)*.

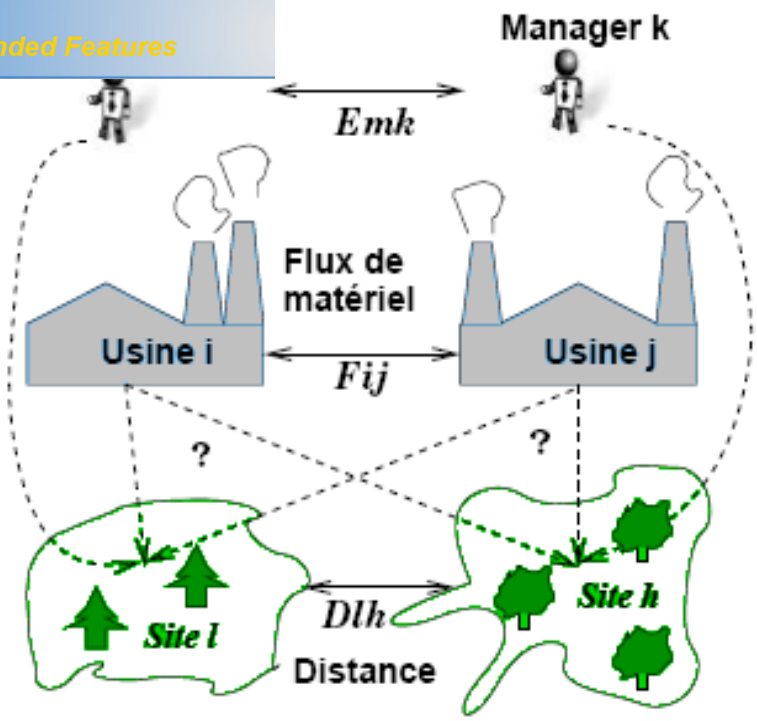

Figure 1.6: Problème d'affectation Quadratique en trois dimensions (Q3AP).

Une présentation détaillée de ce problème fera lôobjet du chapitre 3.

1.4. Les méthodes de résolution

On distingue **[13]** deux grandes familles de méthodes dôoptimisation : les méthodes exactes dédiées à résoudre de façon optimale des instances de petite taille et les méthodes approchées les heuristiques et en particulier les métaheuristiques (génériques) permettant dôapproximer les meilleures solutions sur des instances de grande taille. En effet, la complexité du problème, la nature des variables, des domaines de définition et des critères à optimiser va influencer le choix de la méthode dôoptimisation à utiliser. Nous allons examiner les principales méthodes de résolution qui ont été utilisées pour le problème de l'affectation quadratique. *(cf.Figure 1.7)*

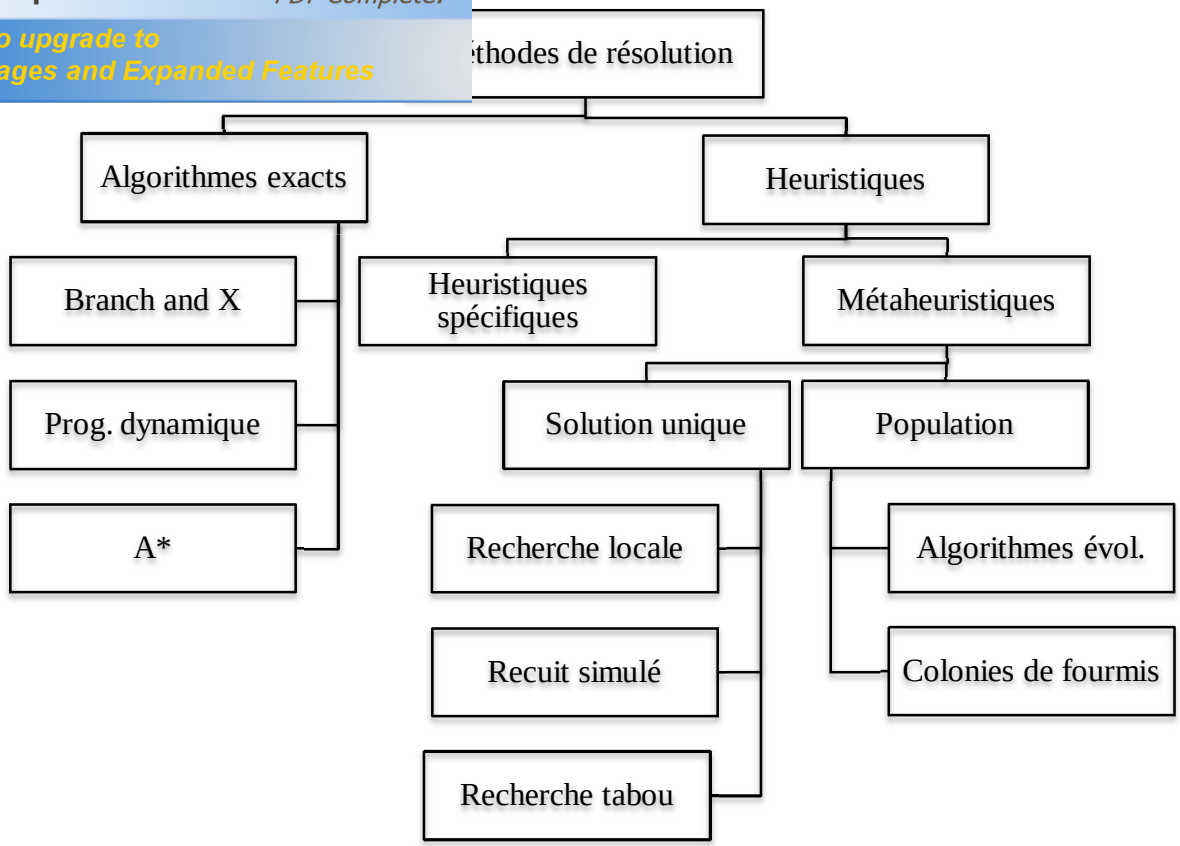

Figure 1.7 : Une taxonomie des méthodes de résolution.

1.4.1. Les Méthodes exactes

Les méthodes exactes cherchent à trouver de manière certaine la solution optimale en examinant de manière explicite ou implicite la totalité de lœespace de recherche. Elles ont lœavantage de garantir la solution optimale néanmoins le temps de calcul nécessaire pour atteindre cette solution devient très excessif en fonction de la taille du problème (explosion combinatoire) et le nombre dœobjectifs à optimiser. Ce qui limite lœutilisation de ce type de méthode aux problèmes de petites tailles. Ces méthodes génériques sont : Branche & bound, Branch & cut et Branch & price, dœautre méthodes sont moins générales, comme : la programmation linéaire en nombre entiers, lœalgorithme de A*. Dœautres méthodes sont spécifiques à un problème donné comme lœalgorithme de Johnson pour lœordonnancement.

Lôalgorithme de branch & bound (procédure par évaluation et séparation progressive), proposé initialement en programmation linéaire en nombres entiers **[14]**, est un grand classique pour résoudre de manière exacte les problèmes d'optimisation. Il s'agit d'une méthode générale, adaptable à de nombreux problèmes combinatoires. On le trouve pour la première fois appliqué au problème MP dans **[15]**. Il consiste à énumérer les solutions dôune manière intelligente en ce sens que, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que lôon recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur lôélimination explicite de toutes les solutions du problème.

Pour ce faire, cette méthode se dote dôune fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure, soit les maintenir comme des solutions potentielles. Bien entendu, la performance dôune méthode de branch & bound dépend, entre autres, de la qualité de cette fonction (de sa capacité dôexclure tôt des solutions partielles).

L'algorithme général

Par convenance, on représente lôexécution de la méthode de branch & bound à travers une arborescence. La racine de cette arborescence représente lôensemble de toutes les solutions du problème considéré. Dans ce qui suit, nous résumons la méthode de branch & bound sur des problèmes de minimisation.

Pour appliquer la méthode de branch & bound, nous devons être en possession:

1. dôun moyen de calcul dôune borne inférieure dôune solution partielle

2. dôune stratégie de subdiviser lôespace de recherche pour créer des espaces de recherche de plus en plus petits.

3. dôun moyen de calcul dôune borne supérieure pour au moins une solution.

La méthode commence par considérer le problème de départ avec son ensemble de solutions, appelé la racine. Des procédures de bornes inférieures et supérieures sont appliquées à la racine. Si ces deux bornes sont égales, alors une solution optimale est trouvée, et on arrête là. Sinon, lôensemble des solutions est divisée en deux ou plusieurs sous-problèmes, devenant ainsi des enfants de la racine. La méthode est ensuite appliquée récursivement à ces sous-problèmes, engendrant ainsi une arborescence. Si une solution optimale est trouvée pour un sous-problème, elle est réalisable, mais pas nécessairement optimale, pour le problème départ. Comme elle est réalisable, elle peut être utilisée pour

la borne inférieure dûun nî ud dépasse la valeur dûune affirmer que la solution optimale globale ne peut être contenue dans le sous-ensemble de solution représenté par ce nî ud. La recherche continue jusquûà ce que tous les nî uds sont soit explorés ou soit éliminés.

b) Branch & cut

Padberg et Rinaldi ont amélioré lûidée du branche & bound basée sur la programmation linéaire en décrivant une méthode utilisant des inégalités renforçant la relaxation par programmation linéaire. Le principe de base est le même que pour le branche & bound, mais au lieu de brancher sur une variable pour descendre dans lûarbre de recherche, on recherche des plans de coupes qui permettent de restreindre lûespace des solutions réalisables **[16]**.

1.4.2. Méthodes approchées

Ces méthodes utilisées pour les problèmes où on ne connaît pas dûalgorithmes de résolution en temps polynomial et pour lesquels on cherche à obtenir une "bonne" solution, sans aucune garantie qu'elle soit la meilleure. Donc, elles sont très utiles pour pouvoir aborder des problèmes de taille plus importante. Elles regroupent les heuristiques spécifiques à un POC particulier et les métaheuristiques. Les premières sont peu réutilisables (les méthodes constructives, gloutonnes, . . .). Par contre, les métaheuristiques sont plus générales et sont indépendantes des POCs traites. Dans ce travail, nous nous intéressons exclusivement aux métaheuristiques.

1.4.2.1. Les heuristiques

Pour certains problèmes, les algorithmes ont une complexité beaucoup trop grande pour obtenir un résultat en temps raisonnable, même si l'on pouvait utiliser une puissance de calcul phénoménale. On est donc amené à rechercher une solution la plus proche possible d'une solution optimale en procédant par essais successifs. Puisque toutes les combinaisons ne peuvent être essayées, certains choix stratégiques doivent être faits. Ces choix, généralement très dépendants du problème traité, constituent ce qu'on appelle une heuristique. Le but d'une heuristique est donc de ne pas essayer toutes les combinaisons possibles avant de trouver celle qui répond au problème, afin de trouver une solution approchée convenable (qui peut être exacte dans certains cas) dans un temps raisonnable. Dans un objectif de résolution de problèmes et de prise de décision, les heuristiques trouvent cependant leur place dans les

tion d'un grand nombre de cas, car celles-ci permettent de examinant d'abord les cas qui ont le plus de chances de donner la réponse.

1.4.2.2. Les Métaheuristiques

Les métaheuristiques ont connu un essor considérable depuis leur apparition dans les années 1970. Elles sont présentées par Osman et Laporte (1996) **[17]** comme étant des méthodes d'approximation conçues dans le but de s'attaquer à des problèmes complexes d'optimisation qui n'ont pu être résolus de façon efficace par les heuristiques et les méthodes d'optimisation classiques. Ces mêmes auteurs définissent formellement la notion de métaheuristique comme étant un processus itératif qui guide une heuristique subordonnée en combinant intelligemment différents concepts pour explorer et exploiter l'espace de recherche, et qui utilise des stratégies d'apprentissage pour structurer l'information dans le but de trouver efficacement des solutions les plus rapprochées possible de la solution optimale. Le développement des métaheuristiques fait partie d'un effort soutenu investi dans le domaine de l'optimisation combinatoire.

Les métaheuristiques les plus classiques sont celles fondées sur la notion de parcours. Dans cette optique, l'algorithme fait évoluer une seule solution sur l'espace de recherche à chaque itération. La notion de voisinage est alors primordiale, les plus connues dans cette classe sont le recuit simulé, la recherche avec tabous, l'autre approche utilise la notion de population elle manipule un ensemble de solutions en parallèle, à chaque itération, on peut citer les algorithmes génétiques, les algorithmes de colonies de fourmis.
Donc on peut classer les métaheuristiques en deux grandes familles: celles à population de solutions et les autres à base de solution unique.

1.4.2.2.1. Métaheuristiques à base de population

Travaillant sur un ensemble de points de lœspace de recherche en commençant avec une population de solution initiale puis de lœaméliorer au fur et à mesure des itérations, caractérisées par une tendance à lœexploration et à la diversification. Ces méthodes sont aptes à générer de « bonnes » solutions, uniformément réparties dans lœespace de recherche. Nous citerons, entre autres, les algorithmes évolutionnaires, les colonies de fourmis et la recherche par dispersion (scatter search).

lutionnaires

Les algorithmes évolutionnaires **[18, 19]** sont basés sur deux principes fondamentaux du processus dôévolution des êtres vivants : la survie des individus les mieux adaptés et la recombinaison génétique. Le principe est de simuler lôévolution dôune population de solutions, en considération de ces deux règles, en vue de converger vers un ensemble de solutions particulièrement bien adaptées à lôenvironnement auquel est identifié le problème traité.

Holland emploie le terme de plan adaptatif (survie des meilleurs individus, élimination des autres), désignant un algorithme dôadaptation dôune espèce dôindividus dans son environnement.

De part sa nature, bien que stochastique, il devrait faire évoluer la population vers des êtres qui y sont de mieux en mieux adaptés.

Les algorithmes évolutionnaires (A.E.) regroupent toutes les techniques stochastiques fondées sur la simulation du processus dôadaptation dans les milieux naturels, et portant les noms dôalgorithmes génétiques **[18]**, de stratégies évolutionnistes [**20, 21]**, de programmation évolutive **[22]** et de programmation génétique **[23]**.

b) Les colonies de fourmis

Les insectes sociaux, comme les fourmis, les abeilles ou les termites sont généralement imaginés dôune manière simple, des animaux non intelligents. Néanmoins, ils exhibent collectivement des compétences impressives pour résoudre les problèmes.

Inspirant de ces insectes, les recherches dans les décennies passées ont guidé en quelques progrès fascinant dans le champ dôalgorithmes naturels. Ces algorithmes imitent la nature dôune manière ou dôune autre. Les réseaux de neurones imitent la structure de notre cerveau humain et les algorithmes génétiques simulent lôévolution. Ils sont caractérisés par le parallélisme inhérent, lôadaptation, le feed-back positif et quelques éléments de lôaléatoire **[24]**.

La recherche guidée par Marco Dorigo produisit un nouveau membre de cette classe dôalgorithmes : lôalgorithme de « le système de fourmis ». Ici, la manière dont ces insectes trouvent le chemin le plus court à partir dôune ressource alimentaire à leur nid est copiée dans une tentative de résoudre les problèmes durs dôoptimisation combinatoire **[24]**. A lôorigine, lôoptimisation par colonie de fourmis a été conçue pour résoudre le problème du voyageur de commerce (PVC) **[25]**.

lus généralement les insectes sociaux, sont des systèmes mplicité de leurs individus, présentent une forte structure dôorganisation sociale. Comme un résultat de cette organisation, les colonies de fourmis peuvent accomplir les tâches complexes qui sont en quelques sortes loin de dépasser les capacités dôindividu pour une seule fourmi.

Lôidée principale côest que les principes de lôauto organisation qui allouent les coordinations de comportement des vraies fourmis peuvent être exploités pour coordonner la population dôagents artificiels qui collaborent pour résoudre les problèmes de lôordinateur. Plusieurs aspects différents pour le comportement de colonies de fourmis ont inspirés des natures différentes dôalgorithmes de fourmis. Des exemples sont le fourrage, division de travailliste, triage de nichée, et le transport coopérative. Dans tous ces exemples, les fourmis coordonnent leurs activités à partir de la stigmergie, une forme depuis une communication indirecte sôinterposa par des modifications de lôenvironnement. Par exemple, une fourmi de fourrage dépose un produit chimique sur le terrain qui monte la probabilité que les autres fourmis suivront le même chemin. Les biologistes ont démontré que quelques comportements de niveaux de colonies dans les insectes sociaux peuvent être expliqué à partir plutôt des modèles simples dont seulement la communication stigmergique est présente. Dans dôautres termes, les biologistes ont démontré quôil est souvent suffisant de considérer la communication indirecte pour expliquer comment les insectes sociaux peuvent atteindre lôauto-organisation. Lôidée derrière les algorithmes de fourmis est alors dôutiliser une forme de stigmergie artificielle pour coordonner les sociétés des agents artificiels **[26]**.

c) La recherche par dispersion

La recherche par dispersion **[27]** est une heuristique évolutionnaire, aujourdôhui répertoriée parmi les méthodes dites de « Path Relinking ». Elle se base essentiellement sur trois opérateurs dits de dispersion, de recombinaison et dôoptimisation. Cette méthode démarre par la génération dôune population initiale de solutions bonnes et bien dispersées dans lôespace de recherche. Lôensemble de référence est alors constitué par la sélection de solutions bonnes et représentatives de la population précédente. Les solutions sélectionnées sont alors recombinées afin de produire de nouvelles solutions exploitées ensuite dans la phase dôoptimisation. Cette dernière consiste généralement en lôapplication dôalgorithmes de descente. Lôensemble de référence mais aussi la population courante sont alors mis à jour à partir de ces solutions « améliorées ». Le procédé est itéré tant quôun critère dôarrêt nôest pas avéré.

à base de solution unique

Les métaheuristiques à base de solution unique reposent sur la notion de recherche de voisinage. Elles démarrent dôune solution initiale (générée aléatoirement ou par une précédente méthode dôoptimisation) puis, de manière itérative, substituent la solution courante par une autre solution, généralement meilleure, de son voisinage candidat. Le processus de recherche cesse lorsquôun certain critère dit de continuation nôest plus avéré **[28]**.

Le schéma dôune métaheuristique basée sur lôamélioration dôune solution unique serait ainsi le suivant.

1. Créer ou sélectionner une solution initiale (par exemple de manière aléatoire), qui devient solution courante,

$$x \in I \xrightarrow[Initialisation]{} S_0 \in R(x)$$

$$S_i \leftarrow S_0$$

2. Générer lôensemble de solutions voisines candidates,

$$S_i \in R(x) \xrightarrow[Voisinage]{} V(Si) \subset R(x)$$

3. Sélectionner un candidat et le substituer à la solution courante,

$$V(S_i) \neq \emptyset \xrightarrow[Sélection]{} S_{i+1}$$

4. Si le critère de décision dôarrêt de la recherche nôest pas vérifié, retourner en 2.

$$S_{i+1} \xrightarrow[Continuation]{} \{Oui, Non\}$$

Les métaheuristiques à base de solution unique sont plus adaptées à intensifier la recherche dans une région prometteuse. Elles regroupent essentiellement les GRASPs **[29]**, les méthodes de recuit simulé **[30, 31, 32]** et de recherche tabou **[33, 34, 35, 36]**.

a) Le GRASP (Greedy Randomized Adaptive Search Procedure)

La métaheuristique GRASP a été proposée initialement par Féo et Resende **[29]**. Côest une méthode multi-départ en deux phases: une phase de construction et une phase de recherche locale ; pour la résolution approchée de problèmes difficiles de lôoptimisation combinatoire.

La Phase de construction : à chaque itération dans cette phase, on construit une liste restreinte de candidats (RCL). Cette liste est constituée par les meilleurs éléments qui ne font pas encore partie de la solution partielle, selon une fonction dôévaluation qui représente pour

pportée par lôincorporation de cet élément, puis on choisit our lôincorporer à la solution en construction.

La phase de recherche locale: cette phase démarre avec la solution résultante de la phase précédente. À chaque itération de sa recherche locale, elle choisit la meilleure solution dans le voisinage de la solution courante. Elle s'arrête jusqu'à atteindre un optimum local *(cf.Figure 1.8).*

```
Procédure GRASP
Pour k allant de 1 à Max_Itération faire
    Solution ← Construction () ;
    Solution ← Recherch_locale (Solution) ;
    Mise_a_jour_Solution (Solution,MeilleurSolution) ;
Fin ;
Retourner MeilleurSolution ;
FinGRASP
```

Figure 1.8 : Pseudo code de GRASP

Lôadaptation de GRASP pour résoudre un problème particulier est assez facile lorsquôils existent des algorithmes de construction et de recherche locale pour ce problème. GRASP a ainsi été appliqué avec succès à un grand nombre de problèmes dôoptimisation comme des problèmes dôordonnancement, de routage ou encore des problèmes théoriques dans les graphes **[37]**.

b) Le recuit simulé

La méthode de recuit simulé s'inspire du processus de recuit physique. Ce processus utilisé en métallurgie pour améliorer la qualité d'un solide cherche un état d'énergie minimale qui correspond à une structure stable du solide. En partant d'une haute température à laquelle le solide est devenu liquide, la phase de refroidissement conduit la matière liquide à retrouver sa forme solide par une diminution progressive de la température. Chaque température est maintenue jusqu'à ce que la matière trouve un équilibre thermodynamique. Quand la température tend vers zéro, seules les transitions d'un état à un état d'énergie plus faible sont possibles *(cf.table 1.1 et Figure 1.3).*

	Problème d'optimisation
	Fonction d'évaluation
Etat d'équilibre	Solution optimale
Rapid quenching	Recherche locale
Température	Paramètre de contrôle T
Careful annealing	Simulated annealing

Tableau 1.1 : Analogie entre le système physique et le problème d'optimisation **[5]**.

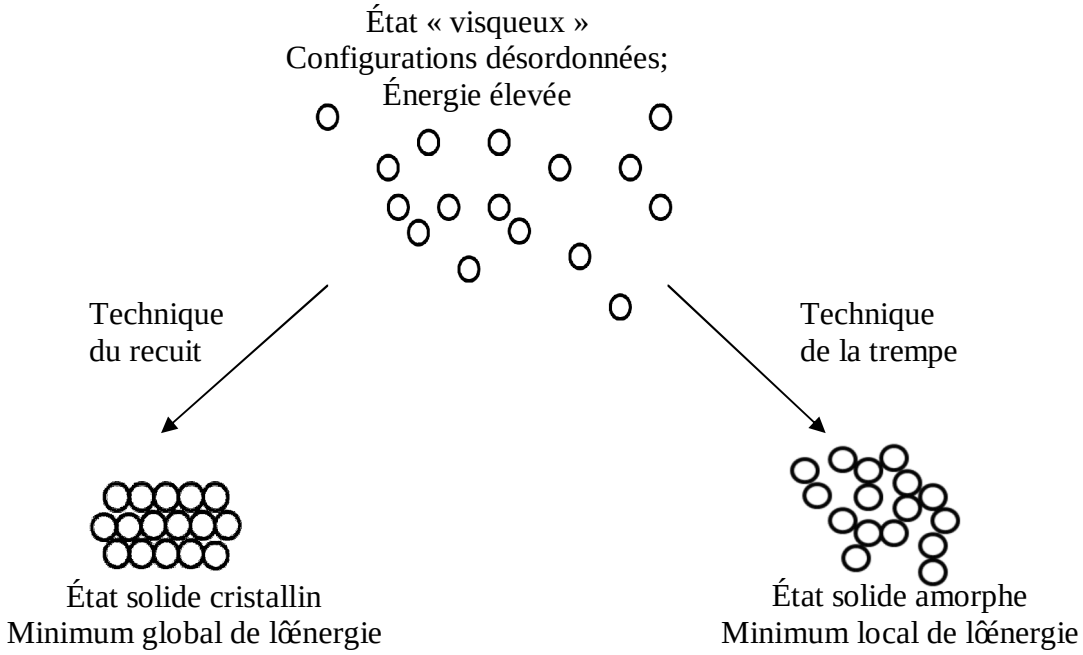

Figure 1.9 : Origine du recuit simulé [38].

Un processus du recuit simulé est utilisé pour résoudre des problèmes d'optimisation combinatoire. Le fonctionnement de cet algorithme est le suivant :

- On commence par choisir un point de départ au hasard.
- On calcule un voisin de ce point (γ = Voisin (x)).
- On évalue ce point voisin et on calcule l'écart par rapport au point d'origine $(\Delta C = C(\gamma) - C(x))$.
- Si cet écart est négatif, on prend le point γ comme nouveau point de départ, s'il est positif on peut quand même accepter le point γ comme nouveau point de départ, mais avec une probabilité $\frac{\square\square\square}{\square\ \square}$ (qui varie en sens inverse de la température T).

…ement de l'algorithme, on diminue la température …(… … (…)), souvent par paliers.

- On répète toutes ces étapes tant que le système n'est figé (par exemple, tant que la température n'a pas atteint un seuil minimal).

Une recherche de recuit simulé acceptera n'importe quelles nouvelles solutions qui sont évaluées comme des solutions supérieures, mais elle acceptera aussi les changements négatifs de qualité avec une probabilité qui dépend de la taille de diminution dans la qualité et la valeur courante de la température. La température commence à une haute valeur, idéalement assez haute que n'importe quelle solution inférieure aura presque une chance de 100 pourcent d'être acceptée, et les diminutions comme un processus exécute ces parcours. La chance d'une solution négative pouvant être acceptée diminue, durant il y a une chance de zéro n'importe quel changement négatif de qualité peut être alloué. A ce point « refroidissement » est dit d'avoir lieu, et le système doit avoir convergé vers la solution optimale **[39]**.

c) La recherche tabou

La méthode de recherche avec tabous a été introduite par Glover (1986) **[33]**. On trouve souvent l'appellation *recherche avec tabous* en français. Cette méthode est conçue en vue de surmonter les minimax locaux de la fonction objectif. Cœst une technique dœoptimisation combinatoire que certains présentent comme une alternative au recuit simulé.

La méthode de recherche tabou consiste à explorer itérativement l'espace des solutions d'un problème en se déplaçant d'une solution courante à une nouvelle solution située dans son voisinage. Le choix de cette nouvelle solution est déterminé par l'évaluation d'une fonction objective **[40]**. Pour sa part, le voisinage d'une solution représente l'ensemble des solutions pouvant être obtenues en appliquant une transformation locale à la solution courante. Afin d'éviter la prise au piège dans un optimum local, une mémoire à court terme est utilisée pour conserver de l'information sur le cheminement récent effectué. Cette information permet d'interdire certaines transformations de la solution courante qui pourraient ramener la méthode à des solutions déjà visitées et qui pourraient la faire cycler indéfiniment dans la même région de l'espace de recherche. Les opérations interdites sont alors déclarées « *tabou* », d'où le nom de la méthode, ce qui a pour effet de restreindre le voisinage aux solutions les plus susceptibles de diriger l'exploration vers des régions inexplorées.

es par ces tabous peuvent parfois être trop contraignantes nant à des solutions non encore visitées. Afin de contrer cette situation, la méthode de recherche avec tabous comporte une fonction d'aspiration qui peut, par exemple, permettre l'annulation du statut tabou d'une transformation locale si cette dernière permet d'obtenir une solution supérieure à la meilleure trouvée depuis le début de l'algorithme.

Bien que cette méthode a montré son efficacité pour la résolution de plusieurs problèmes difficiles, elle reste, néanmoins, difficile à adapter au problème Job Shop flexible. Ceci est surtout dû au nombre considérable de paramètres à définir :

- Solution initiale,
- fonction voisinage,
- évaluation de la solution courante,
- taille de la liste tabou,
- etc.

1.5 Conclusion

Dans ce chapitre, nous avons tenté de dresser un état de lôart sur les problèmes dôoptimisation, tout en montrant la manière de définir un problème pareil, ses contraintes, et ses objectifs. Ainsi quelques exemples des problèmes dôoptimisation combinatoire. Nous avons parlé des méthodes de résolution existantes, nous abordons les notions de méthodes exacte et de méthodes approchés ou les métaheuristique, pour cette raison et puisque notre travail se base sur deux métaheuristiques basées sur une solution nous poursuivrons dans les chapitres suivants en détails ces méthodes.

Chapitre 2

Stratégies pour l'implémentation parallèle des métaheuristiques

Parmi les applications dont la réalisation fait appel au traitement parallèle, les applications de l'intelligence artificielle, dont l'une des branches des métaheuristiques. Ces dernières, comme on a vu dans le chapitre précédent, forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile pour lesquels on ne connaît pas de méthode classique plus efficace, ils sont souvent employés en optimisation combinatoire. Les métaheuristiques sont généralement des algorithmes stochastiques, qui ne nécessitent pas de connaissance particulière sur le problème optimisé pour fonctionner, le fait de pouvoir associer une (ou plusieurs) valeurs à une solution est la seule information nécessaire.

Plusieurs travaux réalisés au cours de ces dernières années ont démontré l'utilité et l'efficacité des métaheuristiques pour la résolution de problèmes d'optimisation combinatoire. Il demeure toutefois que ces algorithmes demandent un temps de calcul et une quantité de mémoire considérables qui sont étroitement reliés à la taille du problème et à l'obtention d'une certaine qualité de solution. De ce fait, ces algorithmes deviennent intéressants à paralléliser. Dans ce contexte, les objectifs de réduction du temps de calcul et de traitement de gros problèmes sont toujours pertinents, mais à cela s'ajoute l'intérêt d'utiliser le parallélisme pour améliorer la performance des métaheuristiques en terme de qualité de solution obtenue et ce, sans augmenter la charge de calcul.

Dans ce chapitre, les principales caractéristiques des métaheuristiques seront brièvement expliquées. Ensuite, une stratégie de parallélisations de métaheuristiques sera présentée.

2.2. Les propriétés fondamentales des métaheuristiques

Les propriétés fondamentales des métaheuristiques sont les suivantes :

- Les métaheuristiques sont des stratégies qui permettent de guider la recherche d'une solution optimale.
- Le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions optimales ou presque.
- Les techniques qui constituent des algorithmes de type métaheuristique vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.
- Les métaheuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité.

t contenir des mécanismes qui permettent dô éviter dô être
espace de recherche.

- Les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, sans faire appel à un problème spécifique.
- Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.
- Les métaheuristiques peuvent faire usage de lô expérience accumulée durant la recherche de lô optimum, pour mieux guider la suite du processus de recherche.

2.3. Stratégies pour l'implémentation parallèle des métaheuristiques

Plusieurs stratégies ont été proposées pour la parallèlisation des métaheuristiques. Ces stratégies prennent en considération les techniques de ces méthodes de recherche, qui différent dô une technique à une autre. Les premiers efforts ont été consacrés pour la parallèlisation du recuit simulé **[41]**, ils ont été suivis par l'implémentation parallèle de l'algorithme génétique et de la recherche tabou.

Pour cette variété de stratégies, il y a plusieurs classifications qui ont été proposées pour classer ces stratégies, par exemple la taxonomie proposée par Verhoven et Arts **[42, 43]**. Ces deux derniers différencient entre les stratégies de marche unique et celles de marche multiple. D'autres classifications répartissent les stratégies en se basant sur une méthode de recherche, comme les algorithmes génétiques (44).

Dans ce qui se suit, nous donnons une description générale de classification des stratégies en se basant sur le chemin. Par la suite, nous essayerrons dô appliquer quelques stratégies de parallèlisation sur la recherche tabou et recuit simulé.

2.4. Classification des stratégies en se basant sur le chemin

La classification des stratégies en se basant sur le chemin a été proposée par Verhoven et Arts **[42]** et elle a été mentionnée par Van-Dat **[43]**. Les métaheuristiques basées sur la recherche locale peuvent être vue comme une exploration de l'espace des états représentés par un graphe avec :

- Les nî uds sont les solutions, où à chaque itération on évolue les solutions dans les voisinages de la solution courante puis on choisit une parmi elles.

ion avec les solutions de voisinage. Le déplacement dôun trajectoire).

Cette classification prend le nombre de chemins utilisés dans l'exploration d'espace des états comme un critère de classification. Le type de ces stratégies différencie entre les stratégies qui utilisent un seul chemin dans l'exploration de l'espace des états et celles qui le font avec plusieurs chemins. Selon le nombre des chemins utilisés dans lôexploration de lôespace des états, on peut distinguer deux approches :

- Exploration avec chemin unique.
- Exploration avec plusieurs chemins.

2.4.1. Exploration avec chemin unique

Dans cette stratégie, il existe un seul chemin à traverser dans l'espace des états, afin dôaccélérer le traitement de l'implémentation séquentiellement. Le parallélisme du chemin apparaît au niveau de la recherche de la meilleure solution voisine de la solution courante: soit par le parallélisme de lôévaluation de la fonction de coût ou par la décomposition du domaine.

a) Parallèlisation de l'évaluation de la fonction de coût

Dans cette stratégie, lôévaluation de la fonction de coût de chaque voisinage de la solution courante est faite en parallèle.

Donc, on distribue les solutions de voisinage sur les différents processeurs pour calculer les valeurs de leur fonction de coût. Le chemin pris par cette méthode est le même que celui dôune implémentation séquentielle, et donc la même solution est trouvée.

b) Parallélisme par décomposition de domaine

Dans cette méthode, à chaque itération de la recherche le domaine de la solution courante est divisé pour le traiter en parallèle. La procédure de cette stratégie a deux phases à chaque itération de la recherche :

- Dans la première phase: Le processeur maître distribue la solution courante avec la décomposition de domaine aux processeurs. Cette décomposition définit le voisinage local (une portion de lôespace de voisinage) pour chaque processeur. La décomposition peut être changée dôune itération à une autre.

aque processeur esclave fait la recherche de la meilleure ocal, puis il la retourne au processeur maître, qui combine les solutions retournées, ou bien il choisit la meilleure solution parmi elles.

La décomposition de domaine est bien détaillée par Aarts et Verheven **[42]**. Ils ont fait la distinction entre le déplacement unique et le déplacement multiple dans chaque sous domaine. Dans le premier cas, dans chaque sous domaine, on fait la recherche et l'évaluation de la solution voisine suivante (un seul déplacement). Dans le deuxième cas, dans chaque sous domaine, on fait une succession de déplacements avant de retourner la solution trouvée dans ce sous domaine *(cf.Figure 2.1).*

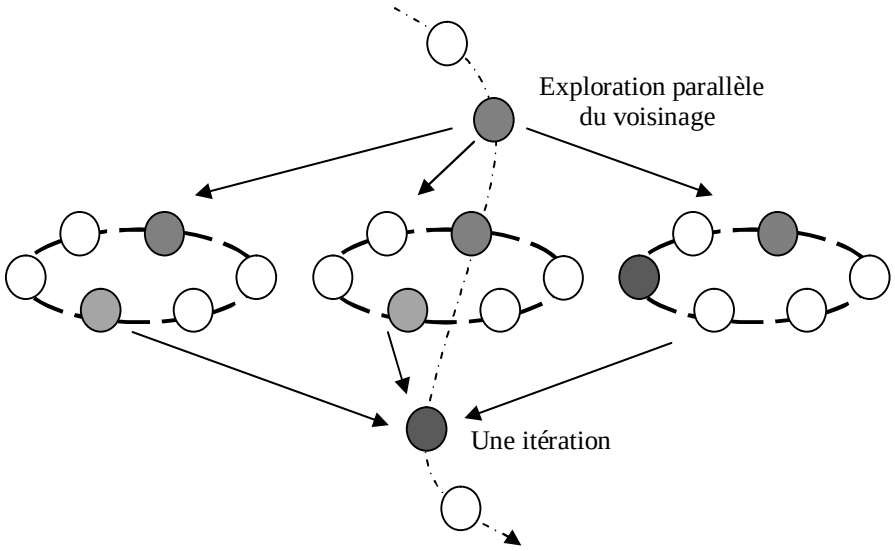

Figure 2.1 : La décomposition et l'exploration parallèle du voisinage d'une solution.

La décomposition de voisinage permet dœxaminer une grande portion de lœspace de voisinage que celui de l'implémentation séquentielle. Donc elle peut donner un chemin différent, et éventuellement elle peut trouver des solutions améliorées. Bien que cette méthode ne donne pas toujours une réduction dans le temps de traitement, mais elle est utilisée pour explorer une grande partie de l'espace des états. Les premières applications de la décomposition de domaine sont apparues dans le contexte du recuit simulé et les algorithmes génétiques **[41]**.

eurs chemins

Dans cette stratégie lôespace des états est exploré par plusieurs chemins en parallèle. Chacun d'eux est pris en charge par un processus (thread). Le travail de ces processus entre eux peut être indépendant (pas de communication entre processus) ou bien coopératif (communication par échange ou par partage dôinformations).

a) Recherche indépendante des processus

Dans la recherche indépendante, il nôy a aucune communication ou partage dôinformations entre les processus pendant la recherche. On peut distinguer deux approches dans cette stratégie *(cf.Figure 2.2)* :

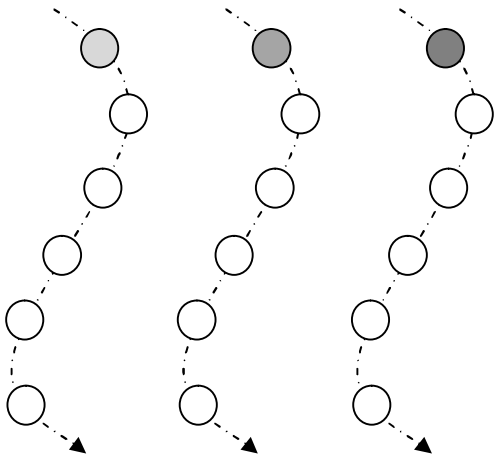

Figure 2.2 : Exploration avec plusieurs chemins indépendants.

➢ ***Exploration en parallèle par des chemins multiples sans division de l'espace des états:***

Chaque processus démarre par une solution différente ou population différente dans le cas des stratégies de population. Lôalgorithme de recherche locale et les valeurs de ses paramètres peuvent être les mêmes ou différentes dôun processus à un autre. Les chemins peuvent être croisés dans un ou plusieurs points au moment de la recherche.

➢ ***Exploration en parallèle par des chemins multiples avec division de l'espace des états:***

Lôespace des états est divisé en sous espaces. Chacun est exploré par un processus en utilisant la méthode de recherche. Dans ce cas, lôexploration de lôespace global des états se fait sans croisement des chemins.

de sur la recherche taboue a montré que les résultats le sont améliorés par rapport à ceux de l'implémentation séquentielle, particulièrement, dans les problèmes de grandes instances.

Ces deux stratégies peuvent être simulées par l'exécution de l'implémentation séquentielle. Par exemple, la première correspond à lôexécution séquentielle de lôalgorithme de recherche répétée le même nombre de fois que les processus utilisés, en changeant à chaque fois le point de départ et les paramètres de la méthode de recherche. La deuxième stratégie, par division de l'espace des états, peut être simulée par l'exécution successive de l'implémentation séquentielle avec différentes initialisations, en utilisant à chaque fois un sous espace de l'espace global des états.

b) Recherche coopérative des processus

La recherche coopérative des processus est considérée comme la plus importante et la plus prometteuse stratégie de parallèlisation. Dans cette stratégie la recherche est faite en parallèle par plusieurs processus. Chacun d'eux explore l'espace des états par la méthode de recherche avec une configuration différente (ou la même) des autres. La coopération entre les processus est faite par partage dôinformations, qui peut être implémentée comme une variable globale accessible par les autres processus, ou par échange dôinformations. Cette information utilisée par chaque processus durant sa recherche sert à améliorer la recherche des autres. Le problème est de déterminer la nature de cette information. Par exemple: les solutions et leurs coûts, les meilleures solutions trouvées, la fréquence de déplacements,

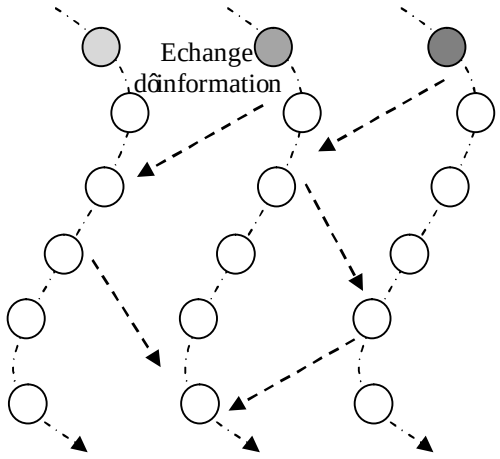

Figure 2.3 : Exploration coopérative avec plusieurs chemins.

opulations. Cette stratégie permet d'obtenir des solutions e la recherche indépendante et avec le même temps d'exécution *(cf.Figure 2.3).*

2.5. Conclusion

Le parallélisme a un grand intérêt pour les métaheuristiques, qui revient au gain en temps de traitement. Plusieurs stratégies sont proposées pour la parallélisation des métaheuristiques, qui sont classées selon plusieurs critères. Parmi ces stratégies, celles qui sont plus convenables avec un type spécifique de métaheuristiques et d'autres qui sont destinées vers une architecture spécifique de machines parallèles. Le parallélisme dûne tâche, dans le sens général, est de la diviser en sous tâches pour les exécuter en parallèle sur différents processeurs, dont le but est de réduire le temps dœexécution et de conserver la fonctionnalité (le but) de la tâche globale. Mais dans le domaine de parallélisation des métaheuristiques, il existe des stratégies qui ne retournent pas forcement le même résultat que l'implémentation séquentielle. Généralement, ce type de stratégies est conçu pour améliorer le résultat de l'implémentation séquentielle, en plus de l'accélération.

Chapitre 3

Recuit Simulé et Recherche Tabou pour la résolution du Q3AP

Dans ce chapitre nous détaillerons toutes les étapes des méthodes ñrecuit simuléò et ñrecherche tabouò pour résoudre le problème dôaffectation quadratique à trois dimensions. Avant expliquer lôapproche utilisée, nous commencerons par une description du problème dôaffectation quadratique à trois dimensions, en passant obligatoirement par un autre problème duquel il a été dérivé. Il s'agit ici du problème d'affectation quadratique simple, QAP. Nous évoquerons à la fois l'aspect formel (formulation mathématique) et l'aspect pratique (modélisation informatique) de ces deux problèmes.

A cause de la complexité de Q3AP et que nous sommes intéressés à la résolution des instances de grand taille, nous avons utilisé deux métaheuristiques parallèle. Dans ce chapitre nous avons donné une nouvelle démarche pour paralléliser les deux métaheuristiques afin dôaméliorer ses performances.

3.2. Problème d'affectation quadratique à trois dimensions

3.2.1. L'origine du Q3AP

Le Q3AP (problème d'affectation quadratique à trois dimensions), est un nouveau problème d'optimisation très difficile, il est présenté en 1967, pour modéliser un protocole de tolérance aux pannes émergeant dans le domaine des télécommunications. Ce protocole, appelé Hybride ARQ (***Hybrid Automatic repeat Request*** ou ***HARQ***) où ARQ désigne un message envoyé lors dôun échange via modem, indiquant la détérioration de certaines informations, le modem demande automatiquement un réacheminement des données abîmées. HARQ consiste á retransmettre le signal original en cas d'erreur détectée, en utilisant á chaque retransmission des codages différents pour lôenvoi de ses codes binaires. Le codage utilisé dans la transmission côest le codage de données QAM (Quadrature Amplitude Modulation) quôest un code spécial utilisé dans la télécommunication pour la transmission des données numérique. Dans l'HARQ, l'idée est d'utiliser des combinaisons symbole QAM - code binaires différents à chaque retransmission de façon à créer une diversité et à minimiser le BER (Bit Error Rate). Donc la modélisation de ce problème de télécommunication a donné lieu à la naissance dôun nouveau problème de recherche quôest le problème dôaffectation quadratique a trois dimensions Q3AP.

parce que l'objet est d'optimiser une fonction quadratique dimensions. La figure 3.1 illustre lôarrangement des coûts dôaffectation dans une matrice cube $N^2 \times N^2 \times N^2$. Il peut être vu comme une extension du célèbre problème dôaffectation quadratique QAP et le problème dôaffectation à trois dimensions (3AP). Par conséquent, le Q3AP est NP-difficile, parce que le QAP et le 3AP, dont elle est une généralisation, sont NP-difficile **[45]**.

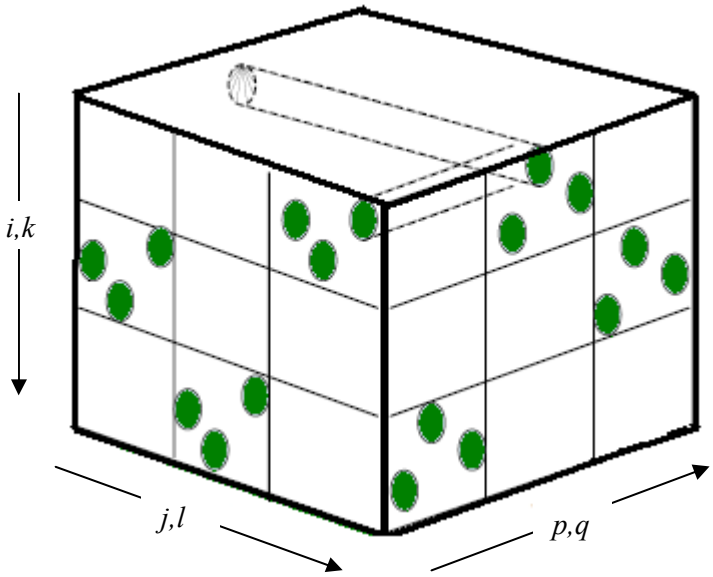

Figure 3.1 : Représentation d'un problème de transmission sans fil comme un Q3AP

3.2.2. Modélisation du QAP et Q3AP

Comme on a vu dans le premier chapitre, le QAP peut définir de manière standard comme une implantation de " n " unités (usines, services hospitaliers...) communiquant entre elles sur " n " sites prédéterminés de façon à minimiser un coût quadratique dépendant à la fois des distances inter-sites et des flux inter-unités. On peut donner une définition standard du Q3AP de manière similaire au QAP, Donc Q3AP consiste à trouver une configuration optimale dôimplantation de " n " unités (usines, services hospitaliers...) communiquant entre elles sur " n " sites prédéterminés et gérer par " n " managers de façon à minimiser un coût quadratique dépendant à la fois des distances inter-sites, des flux inter-unités et des taux dôéchanges inter-managers.

Selon lôorigine du Q3AP et cette modélisation, on trouve que les symboles QAM modélisent par les unités et les manager et les codes binaires modélisent par les sites, de façon suivant *(cf.Figure 3.2)* :

Codes binaires transmet

Symboles QAM

Managers → Symboles QAM

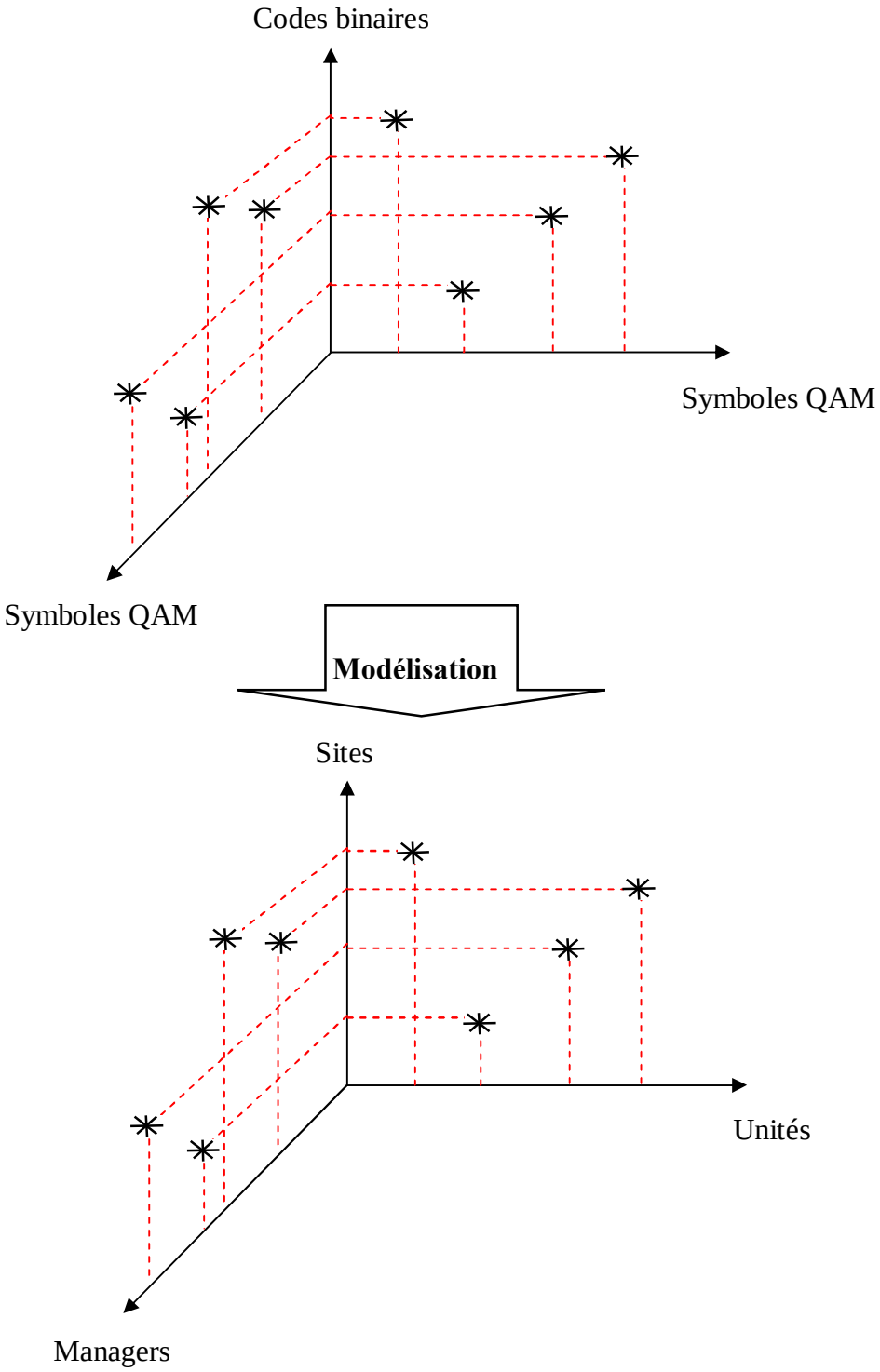

Figure 3.2 : Modélisation du Q3AP.

La formulation de Lawler (1963) du problème d'affectation quadratique est donnée par :

$$\textbf{min} \left\{ \begin{array}{l} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} C_{ijkl}\, x_{ij} x_{kl} + \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij} x_{ij} \\ : x \in I \times J, x \quad \text{binaire} \end{array} \right\} \qquad (1)$$

Où I, J sont des ensembles disjoints et $|I| = |J| = N$

$$\textbf{I=} \left\{ x \geq 0 : \sum_{j=1}^{N} x_{ij} = 1 \ \textbf{pour} \ i = 1,..., N \right\} \qquad (2)$$

$$\textbf{J=} \left\{ x \geq 0 : \sum_{i=1}^{N} x_{ij} = 1 \ \textbf{pour} \ j = 1,..., N \right\} \qquad (3)$$

(b_{ij}) est le coût d'installation de l'unité i sur le site j

Cette formulation est essentielle, car les coefficients objectif {C_{ijkl}} de l'équation (1), pour ces problèmes de communication sans fil ne peut être écrite comme des produits $C_{ijkl} = f_{ik}.d_{jl}$; où (f_{ik}) représenter le flux de matière entre les unités i et k, et (d_{jl}) représenter la distance entre les sites j et l. Les instances du QAP ayant cette dernière propriété sont visées à Koopmans-Beckmann (1957).

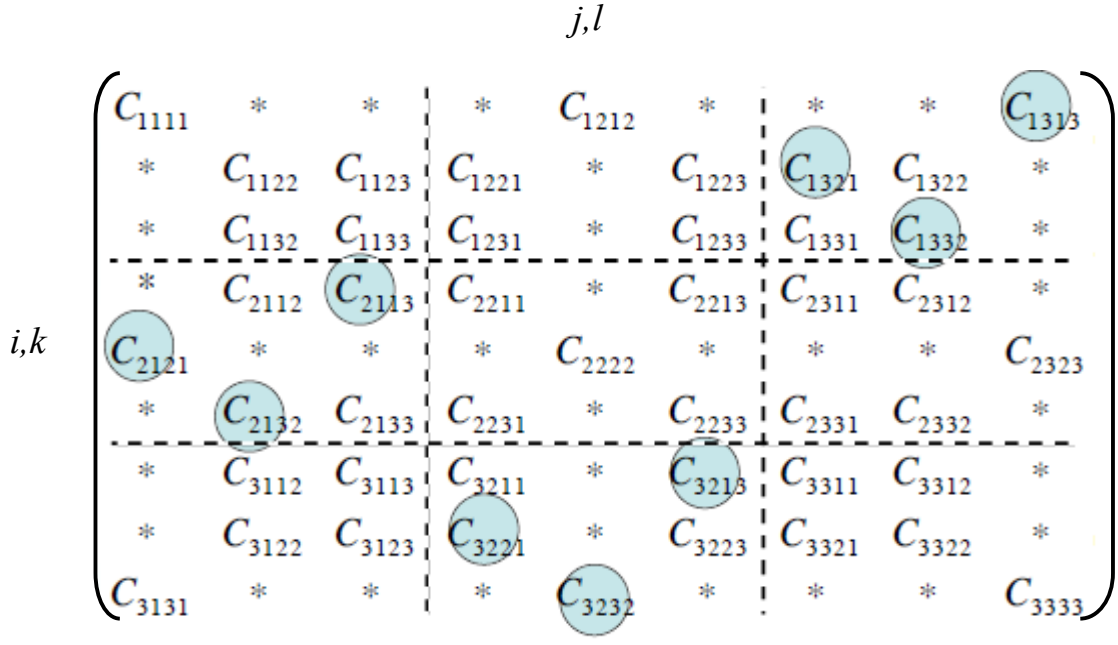

Figure 3.3 : Représentation du QAP de taille 3.

3.3 de Hahn et Grant (1998) **[46]**, on peut placer les coefficients de l'équation (1) dans une matrice $N^2 \times N^2$, tels que les deux premiers indices représentent une sous-matrice d'emplacement et les deux suivants représentent une sous-matrice des différents éléments. Une solution réalisable x à la QAP est une matrice de permutation. On voit facilement que le 1er dans le produit de Kronecker de la matrice de solution sélectionné les coefficients objectifs qui contribuent au coût d'affectation. Ceci est illustré à la figure 3.3 pour une taille 3, où la transparence des disques verts représente les 1ers dans le produit de Kronecker de la matrice de solution *(cf.Figure 3.3)*.

$$\textbf{min} \left\{ \begin{array}{l} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N} b_{ijp}x_{ijp} + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{q=1}^{N} C_{ijpklq}x_{ijp}x_{klq} \\ : x \in I \times J \times P \quad , x \text{ binaire} \end{array} \right\} \qquad (4)$$

Où I, J, P sont des ensembles disjoints et $|I| = |J| = |P| = N$

$$\textbf{I=} \left\{ x \geq 0 : \sum_{j=1}^{N}\sum_{p=1}^{N} x_{ijp} = 1 \quad \textbf{pour} \quad i = 1,..., \ N \right\} \qquad (5)$$

$$\textbf{J=} \left\{ x \geq 0 : \sum_{i=1}^{N}\sum_{p=1}^{N} x_{ijp} = 1 \quad \textbf{pour } j = 1,..., \ N \right\} \qquad (6)$$

$$\textbf{J=} \left\{ x \geq 0 : \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ijp} = 1 \quad \textbf{pour } p = 1,..., \ N \right\} \qquad (7)$$

3.2.4. Formulation du QAP et Q3AP par permutation

Une autre formalisation de ce problème est de représenter une solution sous la forme d'une permutation .Donc, les équations (1) à (3) pour le QAP, peuvent être représentées par:

$$\textbf{min} \left\{ \begin{array}{l} f(\varphi) = \sum_{i=1}^{N} b_{i\varphi(i)} + \sum_{i=1}^{N}\sum_{j=1}^{N} C_{i\cdot\varphi(i)j\cdot\varphi(j)} \\ : \varphi \text{ Permutation de N éléments.} \end{array} \right\} \qquad (8)$$

Q3AP donnée par les équations (4) à (7), peuvent être utations φ et ψ de l'ensemble des entiers {1, ..., N} comme suit :

$$\textbf{min} \left\{ \begin{array}{l} f(\varphi,\psi) = \sum_{i=1}^{N} b_{i\varphi(i)\psi(i)} + \sum_{i=1}^{N}\sum_{j=1}^{N} C_{i\cdot\varphi(i)\varphi(j)\cdot j\cdot\psi(i)\psi(j)} \\ : \varphi;\psi \text{ Permutation de N éléments.} \end{array} \right\} \quad (9)$$

On peut constater facilement que l'augmentation de la taille de l'instance engendre une explosion combinatoire, ce qui rend illusoire une résolution exacte de ce problème.
En effet, les nombre de permutations de {1,..,n} est N! c'est-à-dire, pour un problème d'affectation quadratique de " n " instances on a N! solutions possibles, et pour le Q3AP on a deux permutations donc il y a N!× N! solutions possibles. Il est donc impensable d'envisager de parcourir toutes les possibilités pour en trouver la meilleure. D'où l'omniprésence des heuristiques dans la résolution de ce problème.

3.3. Métaheuristique pour la résolution du Q3AP

3.3.1. Méthodes de voisinage

Une méthode typique de voisinage débute avec une configuration initiale, et réalise ensuite un processus itératif qui consiste à remplacer la configuration courante par l'un de ses voisins en tenant compte de la fonction de coût. Ce processus s'arrête et retourne la meilleure configuration trouvée quand la condition d'arrêt est réalisée. Cette condition d'arrêt concerne généralement une limite pour le nombre d'itérations ou un objectif à réaliser. Un des avantages des méthodes de voisinage réside précisément dans la possibilité de contrôler le temps de calcul : la qualité de la solution trouvée tend à s'améliorer progressivement au cours du temps et l'utilisateur est libre d'arrêter l'exécution au moment qu'il aura choisi. Les méthodes de voisinage diffèrent essentiellement entre elles par le voisinage utilisé et la stratégie de parcours de ce voisinage. Nous verrons dans ce qui suit ces différences dans les méthodes de recuit simulé et recherche tabou.

3.3.2. Méthode de recuit simulé

Le recuit simulé en optimisation combinatoire nôa plus quôun lointain rapport avec la thermodynamique. Lôénergie du système est représentée par un réel arbitraire *T*, la température. A partir dôune recherche locale quelconque pour un problème, on obtient une méthode de recuit comme suit (on part toujours dôune solution réalisable initiale " *s* ") :

- On *tire au sort* une transformation *s* Ÿ *s*ô, *s*ô ∈ *V*(*s*), au lieu de chercher la meilleure ou la première solution voisine améliorante comme dans une recherche locale classique.
- On calcule la *variation de coût* q*f* = *f*(*s'*) ï *f*(*s*).
- Si q*f* Ò 0, le coût diminue et on effectue la transformation améliorante (*s* := *s'*).
- Si q*f* > 0, le coût remonte, côest un rebond, qu'on va pénaliser dôautant plus que la température est basse et que q*f* est grand. Une fonction exponentielle a les propriétés désirées. On calcule une *probabilité d'acceptation* $a = e^{-\Delta f / T}$, puis on tire au sort *p* dans [0,1]. Si *p* Ò *a*, la transformation est déclarée *acceptée*, bien quôelle dégrade le coût, et on fait *s* := *s*ô. Sinon, la transformation est *rejetée* : on garde " *s* " pour lôitération suivante.
-Pour assurer la *convergence*, *T* est diminuée lentement à chaque itération, on peut aussi décroître *T* par paliers. Pour être efficace, un recuit doit diminuer *T* assez lentement, en plusieurs milliers ou dizaines de milliers dôitérations.
- On sôarrête quand *T* atteint un seuil fixé Ů proche de 0.

3.3.2.2. Organigramme

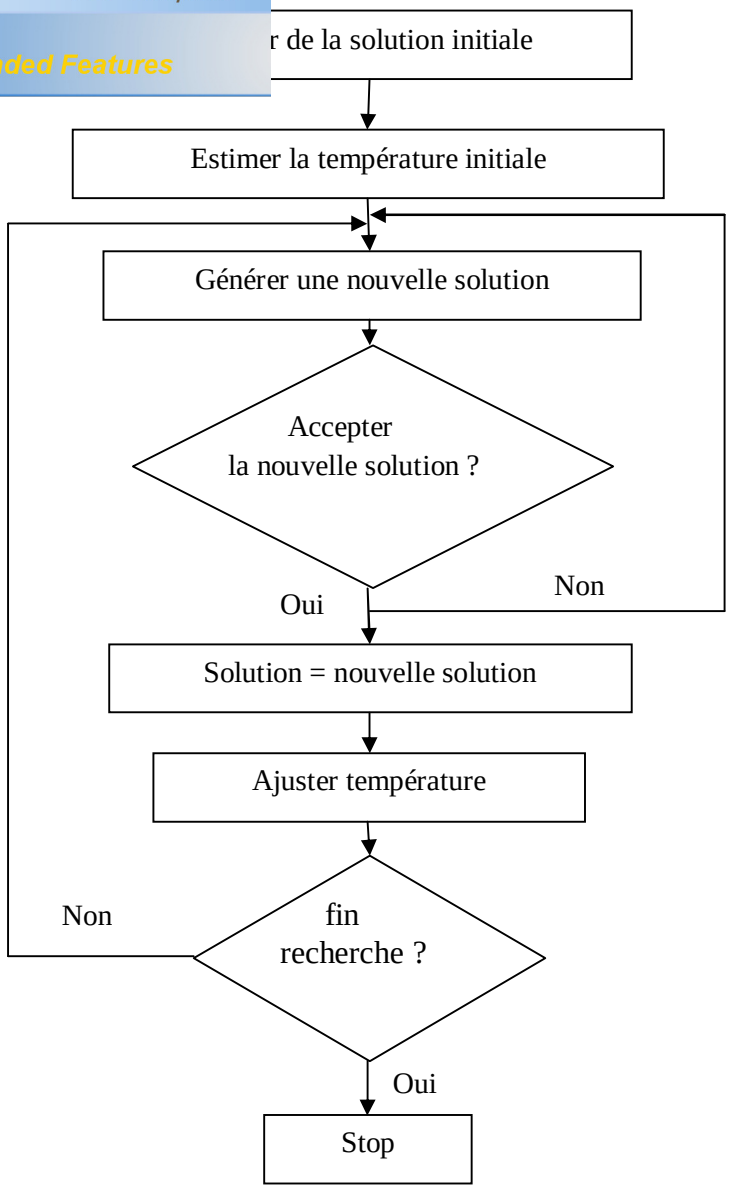

Figure 3.4 : Organigramme de Recuit Simulé.

3.3.2.3. Algorithme

Voici un squelette général de recuit. La solution courante est " *s* ", la meilleure solution trouvée jusquà présent est " *s** ". Les instructions concernant *MaxGel*, *MaxIter* et la sauvegarde de la meilleure solution sont inutiles si le recuit est bien réglé *(cf.Figure 3.5).*

```
r coût obtenu}
s* := s {meilleure solution connue}
Initialiser T et Ů
Initialiser MaxIter et MaxGel
NIter := 0 {nombre dôitérations}
NGel := 0 {itérations sans améliorations}
Répéter
NIter := NIter + 1
Tirer au sort une solution sô dans V(s)
Calculer la variation de coût qpf
Si qpf < 0
Alors Accept := Vrai
Sinon
Tirer au sort p dans [0,1]
Accept := p Ò  e(-qpf/T)
FS
Si Accept alors
s := sô
Si f(s) < z* alors
z* := f(s)
s* := s
FS
Si qpf = 0 alors NGel := NGel + 1 sinon NGel := 0 FS
FS
T := k.T
Jusquôà (T Ò  Ů) ou (NIter = MaxIter) ou (NGel = MaxGel).
```

Figure 3.5 : Algorithme de Recuit Simulé

3.3.2.4. Choix des paramètres

Le réglage des paramètres est assez délicat. Il est prudent de prévoir deux tests dôarrêt supplémentaires : un limitant le nombre dôitérations à une valeur *MaxIter*, et un limitant le nombre dôitérations sans changement de coût à une valeur *MaxGel*. *MaxGel* doit être assez grand : contrairement à une recherche locale, un recuit simulé peut en effet se sortir dôun plateau horizontal par des transformations à coût nul.

En cas de refroidissement trop rapide, la solution finale nôest pas la meilleure trouvée et il vaut donc mieux stocker en cours de route toute solution améliorante.

que pratiquement toutes les configurations proposées soient acceptées.

- Les paliers de température doivent être suffisamment longs pour permettre dôatteindre la distribution stationnaire.

Les lois de décroissance de la température

La loi classique:

$$T_k = \alpha \cdot T_{k-1} \quad (10)$$

Les lois adaptatives:

Van Laarhoven:

$$T_k = T_{k-1} \frac{1}{1 + \frac{\ln(1+\delta)}{3\sigma_{k-1}} T_{k-1}} \quad (11)$$

Huang:

$$T_k = T_{k-1} \exp\left(\frac{-\lambda T_{k-1}}{\sigma_{k-1}}\right) \quad (12)$$

Triki:

$$T_k = T_{k-1}\,\beta - T_{k-1} \frac{1}{\sigma_{k-1}^2} \quad (13)$$

3.3.3. Méthode de recherche tabou

3.3.3.1. Principe

Son principe consiste à explorer lôespace de recherche composé de toutes les solutions réalisables dans le but dôaboutir à la solution optimale et qui minimise la fonction objectif. Commençant à partir dôune solution initiale réalisable, le processus consiste, à chaque itération à choisir la meilleure solution dans le voisinage de la solution courante, même si cette solution nôentraîne pas une amélioration. Il est essentiel de noter que cette opération peut conduire à augmenter la valeur de la fonction : c'est le cas lorsque tous les points du voisinage ont une valeur plus élevée. C'est à partir de ce mécanisme que l'on échappe aux minima locaux. Afin dôéviter le piège des optima locaux dans lequel ce processus peut être facilement attrapé, la recherche tabou utilise une structure de mémorisation temporaire dans laquelle elle sauvegarde les dernières solutions visitée : la liste tabou.

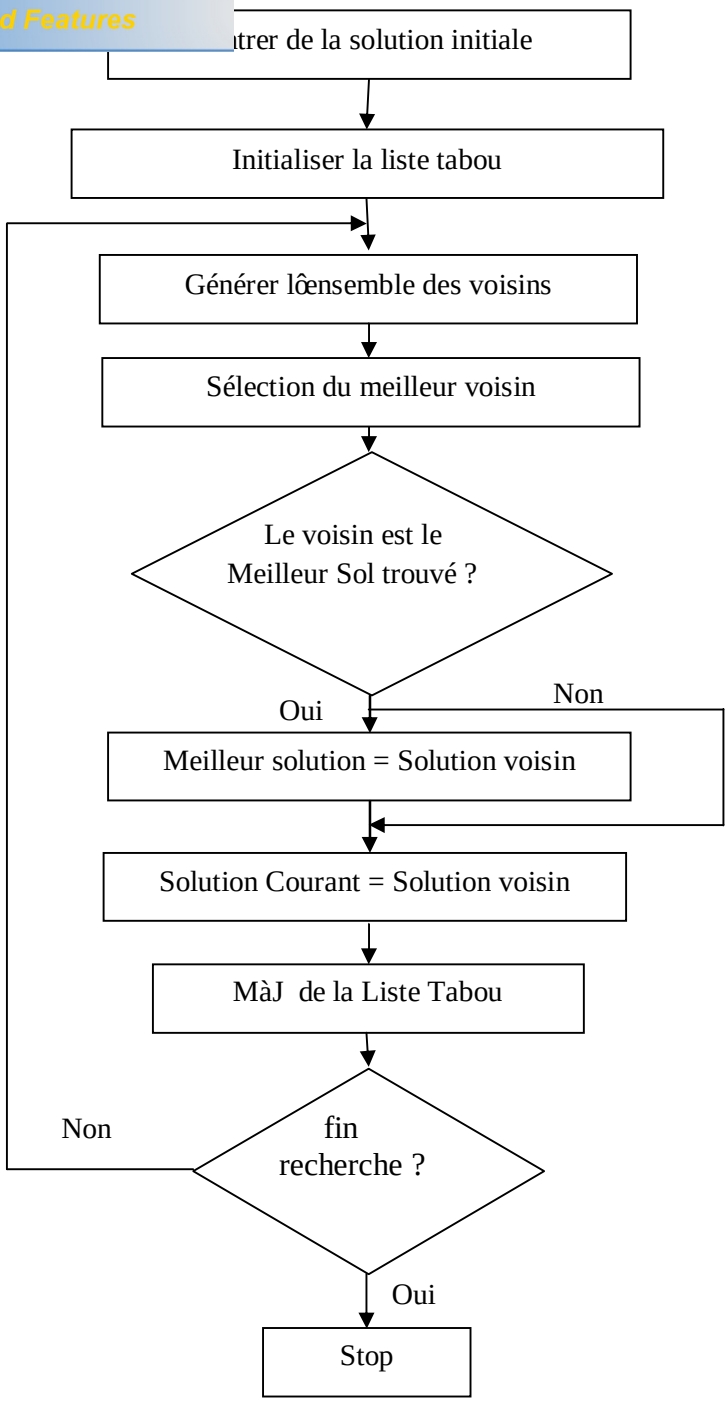

Figure 3.6 : Organigramme de Recherche Tabou.

n squelette général de méthode tabou *(cf.Figure 3.7)*

```
Construire une solution initiale s  {solution initiale}
Calculer son coût z = f(s)
z* := z                             {meilleur coût obtenu}
s* := s                             {meilleure solution connue}
Initialiser MaxIter                 {nombre max d'itérations}
T := ∅                              {vide la liste tabou T}
NIter := 0                          {nombre d'itérations
Répéter                             {boucle principale}
NIter := NIter + 1                  {compte une itération}
z" := +∞                            {initialise coût de s"}
Pour toute solution ŝ de V(s)       {exploration de V(s)}
Si ŝ ∉  T et f(ŝ) < z" alors          {si ŝ n'est pas tabou}
s" := ŝ                             {mise à jour du meilleur voisin s"}
z" := f(ŝ)                          {et de son coût}
FS
FP
Si z" < +∞ alors                    {si s" a été trouvé}
Enlever solution en tête de T       {mise à jour de la liste tabou}
Ajouter s en fin de T
s := s"                             {changement de solution}
z := z"
Si z < z* alors                     {mise à jour meilleure solution s*}
s* := s
z* := z
FS
FS
Jusqu'à (NIter = MaxIter) ou (z" = +∞).
```

Figure 3.7 : Algorithme de Recherche Tabou.

3.3.3.4. Choix des paramètres

La liste tabou T

Glover a montré qu'une liste tabou de taille *NT* = 7 à 20 suffit en pratique pour empêcher l'algorithme de boucler en revenant sur une solution déjà visitée. *T* fonctionne donc comme une sorte de *mémoire à court terme*. A chaque itération, la *NT*-énième solution de *T*

la dernière solution examinée. En pratique, *T* se gère nnées de type *file.*

En théorie, il faudrait stocker complètement les *NT* dernières solutions visitées. Les voisinages considérés étant souvent grands, il est clair que le test répétitif de présence dans *T* est très coûteux, sans parler de la mémoire nécessaire pour coder *NT* solutions complètes. Le stockage explicite des solutions nôest donc *jamais* pratiqué.

En pratique, on stocke dans *T* les *mouvements* ayant permis de changer de solution ou certains *attributs* des solutions. Ensuite, on interdit les mouvements inverses ou les solutions ayant les mêmes attributs. Cette technique est rapide et consomme peu de mémoire, mais elle est plus restrictive que le stockage de solutions complètes. Une technique encore plus simple pour gérer la liste tabou est dôinterdire de repasser par les *NT* dernières valeurs de la fonction-objectif : il suffit de stocker uniquement le coût entier des *NT* dernières solutions. Les résultats peuvent être honorables, sauf si la fonction objectif prend relativement peu de valeurs différentes.

Taille de la liste tabou varie selon les problèmes. Cêst une donnée primordiale :

- Une liste trop petite peut conduire à un cycle.
- Une liste trop grande peut interdire des transformations intéressantes.

Liste tabou dynamique

Lôun des aspects critiques de lôutilisation dôune recherche tabou est la nécessité dôajuster la longueur de la liste tabou pour parcourir efficacement lêspace des solutions. Si la liste est trop courte, la recherche finit par explorer un optimum local de rayon légèrement supérieur. Les différentes solutions explorées forment un cycle qui va se répéter indéfiniment. A lôinverse, si la liste est trop longue, tous les mouvements peuvent devenir tabous et sans nouveaux voisins la recherche sêarrête, cêst un blocage. Le réglage de la longueur de la liste taboue dépend essentiellement de la topologie de lêspace des solutions, pour laquelle le voisinage et le critère tabou sont nos seuls éléments de mesure. En général, la liste tabou doit être maintenue à une longueur minimale permettant dêéviter un cycle. En cas de blocage, tous les mouvements sont tabous. Cette situation est facile à détecter. Il suffit de diminuer la longueur de la liste pour autoriser de nouveaux mouvements.

- « **Stratégie d'intensification** » **:** on mémorise les meilleures solutions et on essaie de dégager des propriétés communes pour définir des régions *intéressantes* (par exemple en rendant tabou tous les mouvements qui font sortir de cette région).
- **« Stratégie de diversification »:** c'est le contraire, on mémorise les solutions les plus fréquemment visitées et imposer un système de pénalités pour favoriser les mouvements les moins souvent utilisés.
- « **Aspiration** »**:** il arrive qu'un mouvement tabou se révèle intéressant. Le critère d'aspiration le plus simple consiste à regarder si le mouvement considéré ne conduit pas à une solution de coût inférieur à celui de la meilleure solution rencontrée

3.4. Application sur Q3AP

3.4.1. Voisinage sur une permutation

Illustrons tout d'abord comment des voisinages simples peuvent être définis pour des problèmes où une solution est une permutation. De nombreux problèmes d'optimisation combinatoire peuvent s'exprimer sous la forme d'une recherche de permutations : celui du voyageur de commerce, celui de l'affectation quadratique comme on a vu précédemment.

A priori, tous ces problèmes pourraient utiliser le même voisinage puisque leurs solutions se représentent sous la même forme. L'inversion de deux éléments contigus est un des voisinages les plus simples pour une permutation. La taille de ce voisinage est en $O(n)$ pour un problème à " n " objets (villes, sites, tâches) et c'est certainement un des plus petits que l'on puisse imaginer fonctionner en pratique (il possède la propriété de connexité pour ces problèmes).

Nous verrons plus loin que les méthodes de recherches locales comme l'affectation quadratique utilisent une structure de voisinage très simple, composée de tous les 2-échanges (ou transpositions), c'est-à-dire de toutes les solutions pour lesquelles la seule différence avec la solution courante consiste en l'échange du placement de deux unités.

Cela est dû tout d'abord parce qu'il ne modifie pas trop la solution, ensuite parce qu'il est rapide à évaluer et enfin parce que sa taille est raisonnable composée de $\frac{?(???)}{?}$ solutions.

n φ est donné par :

$$V(\varphi) = \{\varphi' / \varphi'_r = \varphi_s, \varphi'_s = \varphi_r, r \neq s \ et \ \varphi'_i = \varphi_i \forall i \neq r, s\} \quad (14)$$

Dans le problème d'affectation quadratique à trois dimensions, on a deux permutations φ et ψ de l'ensemble des entiers {1, ..., N}, si on applique le voisinage de 2-échanges sur chacun des deux permutations, on trouve $\frac{N(N-1)}{2}$ solutions possibles qui différent exactement à la solution courant par la position de deux éléments.

$V(\varphi, \psi)$ Voisinage de permutation (φ, ψ) est donné par :

$$V(\varphi,\psi) = \begin{Bmatrix} (\varphi',\psi') / \varphi'_r = \varphi_s, \varphi'_s = \varphi_r, r \neq s \ et \ \varphi'_i = \varphi_i \forall i \neq r,s \ et \ \psi'_i = \psi_i, \forall i \\ ou \ \psi'_r = \psi_s, \psi'_s = \psi_r, r \neq s \ et \ \psi'_i = \psi_i \forall i \neq r,s \ et \ \varphi'_i = \varphi_i, \forall i \end{Bmatrix} \quad (15)$$

Lôéquation (15) peut écrit comme suit :

$$V(\varphi,\psi) = \{(V(\varphi),\psi) \ \ ou \ \ (\varphi, V(\psi))\} \quad (16)$$

Où $V(\varphi)$ et $V(\psi)$ sont des voisinages des permutations φ et ψ respectivement.

A partir de cette dernière équation, on peut avoir que la taille des voisinages est très grande, donc pour le parcours de grande partie des voisinages il faut utiliser le parallélisme.

3.4.2. Parallèlisation de la recherche tabou et recuit simulé

Après lôétude des stratégies de parallèlisation des métaheuristiques, on peut déduire que ces techniques sont autour des trois principales stratégies de parallélisme suivantes :

Å *parallélisme par décomposition de l'espace de recherche* : lôespace de recherche est découpé en sous espaces. Chacun d'eux est exploré par un processus, en utilisant la méthode de recherche, pour but de parcourir une grande portion de lôespace total.

Å *parallélisme par coopération de recherche :* lôespace de recherche est exploré par plusieurs processus en parallèle. Les processus coopèrent entre eux par échange dôinformations pour améliorer et accélérer la recherche.

llèle de la méthode de recherche : une étape ou plusieurs est distribuée parmi les processeurs pour la traiter en parallèle pour but est dôaccélérer la recherche dôune méthode implémentée séquentiellement.

Autour de ces trois stratégies, nous allons construire un nouveau schéma de parallélisme appliqué sur les deux métaheuristiques recuit simulé et recherche tabou pour résoudre le problème dôaffectation quadratique à trois dimensions. Pour la méthode de recuit simulé nous avons fait un petit changement au corps de la méthode, dans lôitération de la méthode on choisit la meilleure solution dans le voisinage de la solution courante au lieu du choix aléatoire, c'est-à-dire lôitération dans les deux méthodes devient la même.

3.4.3. Stratégie de parallélisation proposée

Notre stratégie caractérise principalement par une distribution des calculs, on peut considérer quôil est à deux niveaux de parallélisme quôils sont appliqués à lôintérieur dôune itération des deux métaheuristiques.

Ce modèle de type « Maître / Esclave » dans les deux niveaux, donc on pose « Maître1 / Maître2 » pour le premier niveau « Maître2/ Esclave » pour le deuxième niveau. Comme on a vu précédemment, on a au début du chaque itération une solution « solution courant » composée de deux permutations et au sortie du lôitération on a une autre solution « solution voisine » cette dernière est calculée comme suit :

- **1er Niveau**

Au début de lôitération, le processeur « maître1 » décompose la solution courante en deux solution chacune contient une permutation qui sera émise vers différents processeurs, lesquels considéreront un voisinage partiel de solutions candidates. Les « maître2 » explorent de façon concurrente et indépendante l'espace de recherche qui leur a été assigné par le « maître1 » et ce dernier assemble les solutions partielles trouvées par les « maître2 » en une solution complète au problème avant de poursuivre sa marche *(cf.Figure 3.8)*. On remarque que lôespace de solution est divisé en deux sous espaces, chaque processeurs « maître2 » est utilisé sur lôun de ces sous espaces. Comment lôalgorithme « maître2 » va trouver les voisins candidats côest le problème du deuxième niveau.

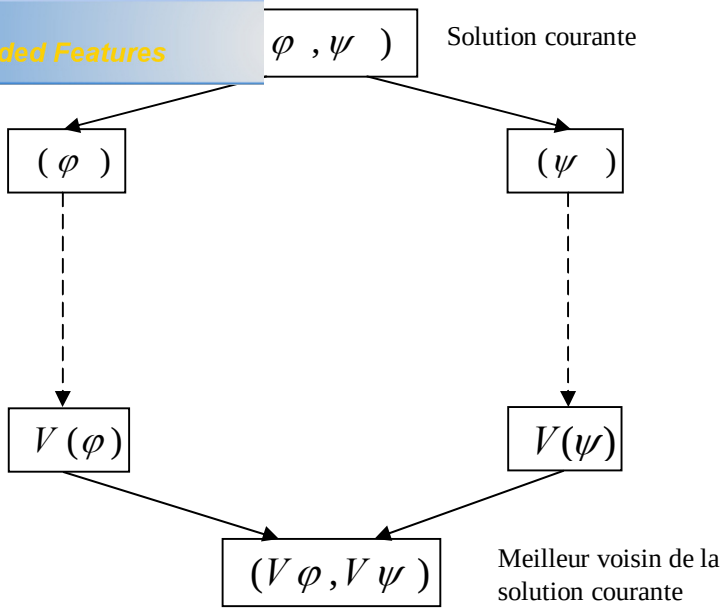

Figure 3.8 : Schéma du 1er niveau de la stratégie de parallélisation proposée.

➢ 2ème Niveau

Dans ce niveau, on a deux itérations qui sont traitées en parallèle par les processeurs « maître2 », à chaque itération de la recherche le domaine de la solution courante est divisé pour le traiter en parallèle aussi. La procédure de ce niveau a deux phases à chaque itération de la recherche comme dans le premier niveau

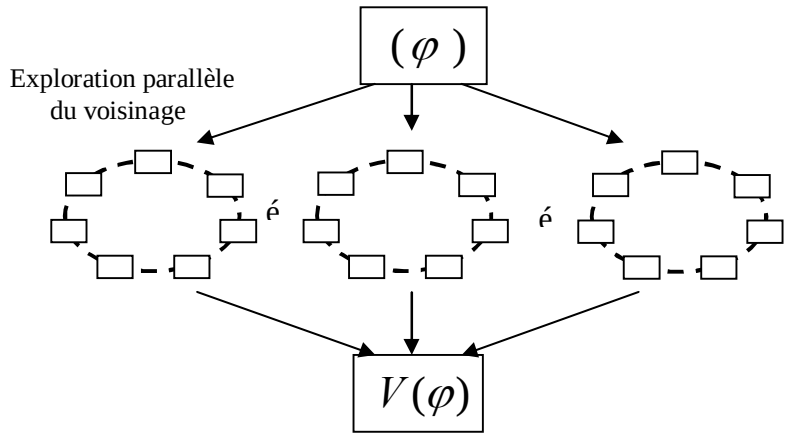

Figure 3.9 : Schéma du 2ème niveau de la stratégie de parallélisation proposée.

processeur « maître2 » distribue la solution courante avec ine aux processeurs. Cette décomposition définit le voisinage local (une portion de lôespace de voisinage) pour chaque processeur. La décomposition peut être changée dôune itération à une autre.

- Dans la deuxième phase: Chaque processeur « esclave » fait la recherche de la meilleure solution dans son voisinage local, puis il la retourne au processeur « maître2 », qui choisit la meilleure solution parmi elles.

A la fin de deuxième niveau de notre stratégie, on trouve deux meilleurs voisins de chaque permutation de la solution courante, quôils retournent au processeur « maître1 », qui combine les solutions retournées, afin de trouver une meilleure solution voisin de la solution courante.

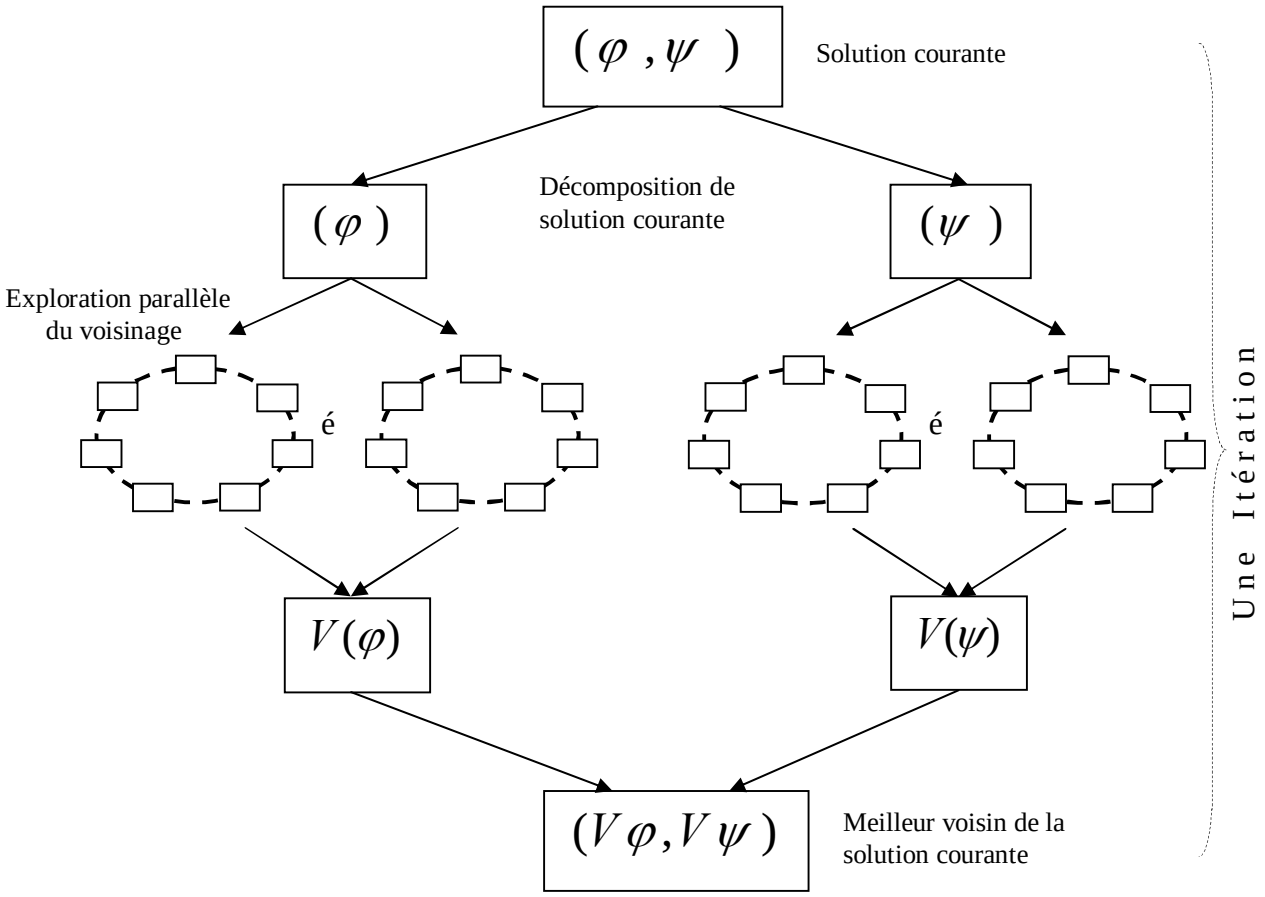

Figure 3.10 : Schéma globale de la stratégie de parallélisation proposée.

ition de la solution voisine est donnée par lôéquation (15) est changée de manière suivante :

$$V(\varphi,\psi)=\begin{Bmatrix}(\varphi',\psi')/\varphi'_r=\varphi_s,\varphi'_s=\varphi_r,r\neq s\ et\ \varphi'_i=\varphi_i\forall i\neq r,s\\ et\ \psi'_u=\psi_v,\psi'_v=\psi_u,u\neq v\ et\ \psi'_i=\psi_i\forall i\neq u,v\end{Bmatrix} \quad (17)$$

Au lieu de faire un changement dans lôun des deux permutations nous avons fait dans les deux en même temps, donc lôéquation (16) devint :

$$V(\varphi,\psi)=\{(V(\varphi),V(\psi))\} \quad (18)$$

Cette stratégie est appliquée sur les deux métaheuristiques recuit simulé et recherche tabou afin de résoudre le problème dôaffectation quadratique à trois dimensions.

3.5. Conclusion

Voilà dans ce chapitre, nous avons présenté un état de lôart sur deux problèmes QAP et Q3AP dôune manière détaillé, ensuite nous avons présenté deux méthodes approchées pour le résoudre qui sont la méthode de recuit simulé et la méthode de recherche tabou. Les résultats retournés par les deux méthodes sur les paramètres de cette dernière. Ces paramètres sont constituées essentiellement par: la température et sa manière de décroissement pour la méthode de recuit simulé et la liste tabou pour la méthode de recherche tabou.

Pour améliorer le résultat de l'implémentation séquentielle des métaheuristiques, en plus de l'accélération, nous avons introduit une nouvelle stratégie de parallélisation qui est inspirée des stratégies existantes.

Chapitre 4

Implémentation

Le but de notre travail nôest pas de trouver la solution optimale par l'implémentation séquentielle. Mais, le but est dôaccélérer le traitement en premier lieu et de conserver la valeur de la solution à retourner de ne pas la dégrader ou de l'améliorer à celle de l'implémentation séquentielle en deuxième lieu.

Dans ce chapitre, nous présenterons la plateforme ParadisEO utilisé dans ce travail qui intègre lôensemble des métaheuristiques que nous avons étudié. Puis nous définirons la bibliothèque QAPLIB des instances utilisé. A Cause du manque des instances pour le Q3AP nous avons faire une implémentation sur le QAP.

Afin de nous permettre de comparer les deux méthodes, nous présenterons les caractéristiques et les résultats de lôapplication de nos métaheuristiques parallèles au problème dôaffectation quadratique et citerons quelques résultats trouvés dans la littérature des applications des deux méthodes sur Q3AP.

4.2. ParadisEO (PARAllel and DIStributed Evolving Objects)

Une plate-forme pour la conception et le déploiement de métaheuristiques parallèles ParadisEO constitue une extension de la plateforme EO, développée dans le cadre dôun projet Européen, et initialement dédiée à la mise en î uvre dôalgorithmes évolutionnaires séquentiels. Nos contributions sôadressent principalement au support des métaheuristiques à base de solution unique, lôhybridation dôalgorithmes, la conception et lôintégration de nouveaux mécanismes appliqués à lôoptimisation multi-objectif et enfin le calcul paralléle et distribué. Basée sur une séparation conceptuelle claire entre les métaheuristiques et les problèmes à traiter, ParadisEO fournit la partie invariante des métaheuristiques et laisse à lôutilisateur le soin de fournir la partie dépendante de son problème. Ce dernier passera ainsi plus de temps sur la modélisation de son problème que sur la méthode de résolution de celui-ci. La partie du code spécifique au problème sera appelée à lôexécution par les composants logiciels fournis suivant le principe de Hollywood « Ne nous appelez pas, nous vous appelons » (Do not call us, we call you).

A la différence de beaucoup dôautres plateformes existantes, ParadisEO intègre un large éventail dôopérateurs et de mécanismes réutilisables. Tout dôabord, elle couvre les méthodes évolutionnaires ainsi que celles à base de recherche locale. Ensuite, pour chacune de ces classes, elle offre une gamme étendue dôopérateurs génériques (e.g. opérateurs génétiques, stratégies dôexploration du voisinage, critères de décision de continuation, . . .).

rallèles sont implémentés, utilisables de manière simple,
es sur des environnements dœexécution paralléles et/ou distribués dédiés.

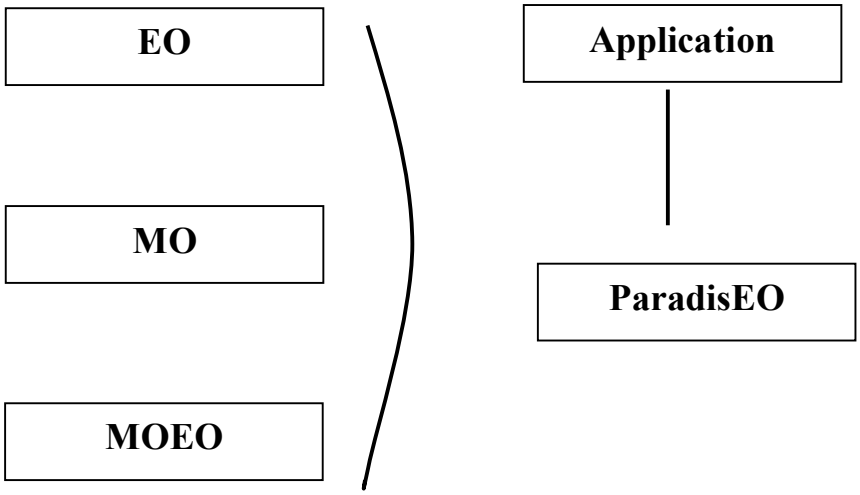

Figure 4.1 : Une organisation modulaire du projet ParadisEO

ParadisEO secompose basiquement de trois modules indépendants et complémentaires désignés par« Evoloving Objects »(Evolutionary Algorithms) (EO) « Moving Objects »(Local Seaches) (MO), « Multi-Objective Evolving Objects » (Muti-objective Optimization)(MOEO) *(cf.Figure 4.1)*. Nos contributions concernent la conception de métaheuristiques à base de solutions uniques (Méthodes de recuit simulé et recherche Tabou), donc nous intéressons au module MO. (28)

Principales caractéristiques

- Une librairie Open Source C++ (STL-Template)
- Indépendante de tout paradigme,
- Flexible / problème traité
- Composants génériques (opérateurs de recherche, sélection, remplacement, terminaison, é)
- Nombreux services (visualisation, définition de paramètres en ligne, sauvegarde/reprise de lœétat dœexécution, é)

Burkard, Karisch et Rendl **[47]** ont regroupé plusieurs instances dans QAPLIB Cette librairie permet d'avoir les informations suivantes sur chacune de ces instances :

- Nom de l'instance : ce nom se décompose sous la forme d'un radical de trois lettres mis pour les trois premières lettres du nom du premier auteur. Le suffixe est constitué d'une valeur qui représente la taille du problème. Par exemple, l'instance had20 a été crée par S.W. Hadley, F. Rendl et H. Wolkowicz, et correspond à un problème contenant 20 usines et 20 emplacements.
- La solution optimale quand elle est connue.
- Pour les instances dont on ne connaît pas la solution optimale, on donne la meilleure borne inférieure et la meilleure solution approchée.

Afin d'évaluer les performances de notre approche, nous avons effectué des tests numériques sur différentes instances du QAPLIB 4.1.

4.4. Les ressources utilisées

Les ressources utilisées pour réaliser ce travail sont :

1- Un micro-processeur Pentium IV de vitesse 3.40 GHz,

2- Une RAM de 1 Go,

3- Le système dœxploitation XP professionnelle service Pack 2,

Nous avons implémenté nos algorithmes en langage C++.

4.5. Résultats de l'application sur QAP

Nous avons testé les deux méthodes recuit simulé (RS) et recherche tabou (RT) parallèle sur plusieurs instances du QAP de taille 20, 30, 40, 50, 80 et 100. Pour chacun des deux algorithmes, on cherche à trouver la solution optimale pendant quelques secondes de temps de calcul et sa pour les instances du tailles 20,30 et 40. Les résultats ne sont pas indiqués explicitement ici. Les résultats expérimentaux sur les instances de taille 50,80 et 100 sont résumés dans les tableaux (tableau 4.1, tableau 4.2 et tableau 4.3). Nous avons donné les valeurs moyennes de la fonction objectif de lœxécution de RS et RT sur les instances (tai50a, tai80a, tai100a) utilisant des délais différents de 1minute à 10 heures.

		10minutes	1 heure	10 heures
RT parallèle	6500540	6074589	5501230	**4941410 ***
RS parallèle	6298441	5994500	5791407	**4941410 ***

*Valeur optimale de la fonction objectif

Tableau 4.1 : Les valeurs moyennes de la fonction objectif de l'exécution de RS et RT sur les instances tai50a

	1minute	10minutes	1 heure	10 heures
RT parallèle	19526696	18455560	15871301	**13511780***
RS parallèle	18914777	17965322	15990037	13526696

Tableau 4.2 : Les valeurs moyennes de la fonction objectif de l'exécution de RS et RT sur les instances tai80a

	1minute	10minutes	1 heure	10 heures
RT parallèle	29456545	28722710	23901005	**21052466***
RS parallèle	29012488	28166014	24979900	21071558

Tableau 4.3 : Les valeurs moyennes de la fonction objectif de l'exécution de RS et RT sur les instances tai100a

Selon les résultats indiqués dans les tableaux précédents, nous concluons que la méthode de recherche tabou parallèle est meilleur que la méthode de recuit simulé parallèle, si on applique sur le QAP, parce que avec le premier nous avons trouvé lôoptimum exact comme indiquer dans QAPLIB dans les trois instances qui utilisent dans notre test. Pour la méthode de recuit simulé parallèle, nous ne trouvons pas lôoptimum dans le cas des instance du taille 100.

Mais ces résultats nôindiquer pas forcément la performance de la recherche tabou si on applique sur Q3AP, parce que la stratégie de parallélisation, que nous avons proposé, est

P ou on a deux permutations, et pour le QAP on peut la stratégie.

Si pour cela que nous avons présentés quelques résultats dôapplication des deux méthodes sur le Q3AP présentés par Hahn et al **[48]**.

4.6. Quelques résultats de l'application sur Q3AP

Hahn et al. appliquent plusieurs méthodes de résolution sur le Q3AP, nous avons intéressé aux deux méthodes recuit simulé (RS) et recherche tabou (RT) qui testent sur plusieurs instances du Q3AP de tailles 12, 13 et 15. Les résultats trouvés sont résumés dans les tableaux (tableau 4.4, tableau 4.5 et tableau 4.6). Les deux méthodes sont implémentées séquentiellement.

	1minute	10minutes	1 heure	10 heures
RT	779	748	607	**580***
RS	771	705	627	583

*Valeur optimale de la fonction objectif

Tableau 4.4 : Les valeurs moyennes de la fonction objectif de l'exécution de RS et RT sur les instances Nug12

	1minute	10minutes	1 heure	10 heures
RT	2118	1995	1918	**1912***
RS	2102	2024	1967	**1912***

Tableau 4.5 : Les valeurs moyennes de la fonction objectif de l'exécution de RS et RT sur les instances Nug13

		10minutes	1 heure	10 heures
RT	2843	2642	2547	2324
RS	2752	2631	2504	2400

Tableau 4.6 : Les valeurs moyennes de la fonction objectif de l'exécution de RS et RT sur les instances Nug15

Les résultats de Hahn et al. *(cf. tableau 4.4, tableau 4.5 et tableau 4.6)* certifient que la méthode de recherche tabou est meilleure que la méthode de recuit simulé pour le Q3AP. En attente des résultats d'application des méthodes recherche tabou et recuit simulé qui parallélisent selon la stratégie proposée dans ce travail, sur le Q3AP, on peut accepter que la méthode de recherche tabou est la meilleure.

4.7. Conclusion

La plate-forme logicielle utilisée est une plate-forme pour la conception et le déploiement des métaheuristiques parallèles, l'intérêt des métaheuristiques pour le Q3AP réside dans la résolution du problème s'il s'agit des instances de grandes tailles. Le besoin d'accélérer le traitement n'est pas dû seulement à la grande taille des instances mais aussi à l'utilisation d'une configuration difficile pour les métaheuristiques comme l'augmentation du nombre des itérations pour but d'améliorer la qualité. Donc, le traitement parallèle peut nous aider pour améliorer les performances des métaheuristiques.

Tous les résultats trouvés dans la littérature et qui nous avons trouvé par l'application sur le QAP montrent que la méthode de recherche tabou et plus performante que la méthode de recuit simulé.

Conclusion et perspectives

Le but de toute méthode de recherche est de trouver la solution optimale au bout d'un temps déterminé ou dans le plus court temps. La qualité de la solution retournée et le temps de traitement sont liés par le type du problème à résoudre. En plus de la difficulté du problème, la taille des instances joue un rôle important en temps de traitement. Lorsque la taille des instances du problème est importante, le temps d'évaluation d'un état ou de trouver les états suivants pendant la recherche devient considérable.

La plupart des problèmes rencontrés dans le monde de la recherche opérationnelle sont NP-difficiles, ce qui ne nous permet pas d'avoir des méthodes exactes pour les résoudre. Il convient dès lors de renoncer à l'idée de la détermination d'une solution optimale et se contenter seulement de chercher une bonne solution, en un temps raisonnable par l'utilisation d'heuristiques ou des métaheuristiques. Cependant, l'utilisation de ces dernières nécessite, non seulement une bonne définition et modélisation du problème à traiter, mais aussi une bonne détermination des paramètres de chacune d'elles. Toutefois, le choix des méthodes à appliquer constitue en lui même une problématique.

Dans ce mémoire, nous avons tenté de résoudre l'un des problèmes d'optimisation combinatoire le plus difficile, le problème d'affectation quadratique à trois dimension par deux métaheuristiques parallèles « recuit simulé et recherche tabou ». Dans ce contexte, les objectifs de réduction du temps de calcul et de traitement de gros problèmes sont toujours pertinents, mais à cela s'ajoute l'intérêt d'utiliser le parallélisme pour améliorer la performance des métaheuristiques en termes de qualité de solution obtenue et ce, sans augmenter la charge de calcul.

Pour cela nous avons étudié les différentes stratégies de parallélisation des métaheuristiques, ensuite nous avons fait une analyse sur la nature des méthodes utilisent dans ce travail qu'ils sont les deux de nature itérative, puis nous avons défini les variables du problème, finalement nous avons construit une nouvelle stratégie qui dépend de tous les paramètres précédents.

A Cause du manque des instances pour le Q3AP nous avons implémenté sur les instances du QAP. Puis, nous avons présenté quelques résultats qui se trouvent dans la littérature des applications des deux méthodes sur Q3AP. Afin de nous permettre de

comparer les deux méthodes. Après ces résultats nous concluons que la méthode de recherche tabou est toujours meilleure et plus performent que la méthode de recuit simulé.

Finalement, l'hybridation entre est deux méthodes étudié « recuit simulé et recherche tabou » est en perspective de ce travail. Et l'hybridation entre les méthodes exactes et les méthodes approchées c'est une autre voie de recherche.

[1]. **Rémy-Robert, Alexandre Joseph.** Systèmes Interactifs dôAide à lôÉlaboration de Plannings de Travail de Personnel Contraintes, Aide à la Décision, Représentation Combinatoire des Préférences, Équité et Résolution par Décomposition Arborescente et par Consistance. 07 Novembre 2003. Université Joseph Fourier-Grenoble1, Science et Géographie ; Page 28.

[2]. **LEILA.SAADI.** Optimisation MultiObjectifs par Programmation Génétique. thèse magistère. 08 juillet 2007 : s.n. Université de BATNA.

[3]. **Alain.Berro.** Optimisation Multiobjectif et Stratégie dôévolution en. 18 Décembre 2001. Université des sciences sociales ToulouseI ; page 14, 27, 29.

[4]. **Dorian.Gaertner.** Natural Algorithms for Optimisation Problems. Final Year Project Report. June 20, 2004.

[5]. **Elghazali.Talbi.** Méthodes dôOptimisation Avancée. Page 18, 19.

[6]. **Johnson.Michael, R. Garey and David S. Johnson.** Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman & Company, San Francisco,1979.

[7]. **Frédéric Lardeux, Frédéric Saubion et Jin-Kao Hao.** Une étude empirique des heuristiques de branchement pour le problème MAX-SAT. LERIA, Université dôAngers, 2 Bd Lavoisier, 49045 Angers Cedex 01: 2006.

[8]. **A. Ramani, F. A. Aloul, I. L. Markov et K. A. Sakallah.** Breaking InstanceIndependent Symmetries in Exact Graph Coloring. 2004; Page 2.

[9]. **G.J.Chaitin, et al.** Register allocation via coloring; in Computer Languages. 6:47-57; 1981.

[10]. **Riven. I, Vardi. I, and Zimmermann, P.** The n -Queens Problem. Amer. Math. Monthly 101, 629-639, 1994.

[11]. **Beckmann, T.C. Koopmans and M.J.** Assignment Problems and the Location of Economic Activities. Econometrica, vol. 25, pp. 53-76, 1957.

[12]. **Pierskalla, W.P.** The Multi-Dimensional Assignment Problem. Technical Memorandum No.93. Operations Research Department, CASE Institute of Technology, September 1967.

thermodynamically motivates simulation procedure for ... European journal of Operational Research,17 :169-174, 1984.

[14]. **Ailsa H. Land, et Alison G.Doig.** An automatic method of solving discrete programming problems. Econometrica, 28(3):497{520, 1960.

[15]. **Michael D. Hendy, David Penny.** Mathematical Biosciences. 60:133{142, 1982.

16. **Thiene, (Stefan).** ï ABACUSð A Branch-And-CUt System. Thèse de PhD, Universit¨at zu K¨oln, 1995.

[17]. **I.H.Osman, and G.Laporte.** Metaheuristics: a bibliography. Annals of Operations Research 63, 513-623, 1996.

[18]. **J.H.Holland.** Adaptation in natural and artificial systems. Ann Arbor, MI, USA, The University of Michigan Press, 1975.

[19]. **D.Goldberg.** Genetic algorithms in search optimization and machine learning. Addison-Wesley, 1989.

[20]. **I.Rechenberg.** Evolution Strategie : Optimierung technischer Systems nach Prinzipien der bilogischen Evolution. Fromann-Holzboog Verlag, Stuttgart, 1973.

[21]. **H-P.Schwefel.** Numerical Optimization of Computer Models . Wiley, Chichester,1981.

[22]. **L.J.Fogel, A.J.Owens, and M.J.Walsh.** Artificial Intelligence Through Simulated Adaptation. Wiley, New-York, 1966.

[23]. **J.R.Koza.** Genetic Programming. Bradford / MIT Press, 1994.

[24]. **D.Gaertner.** Natural Algorithms for Optimisation Problems. Outsourcing Report; 16 Janvier 2004.

[25]. **C.Gagné, M.Gravel et W. L.Price.** Optimisation par Colonie de Fourmis pour un Problème dôOrdonnancement Industriel avec un Temps de Réglage Dépendant de la Séquence. 3ème Conférence francophone de Modélisation et Simulation « Conception, Analyse et Gestion de Systèmes Industriels » ; MOSIMô01, Troyes, France ; 2001.

[26]. **M.Dorigo.** From Real to Artificial Ants. Chapitre 1 ; 1998.

[27]. **F.Glover.** Heuristic for integer programming using surrogate constraints. Decision Sciences, 8 :156ï 166, 1977.

[28]. **Sébastien, CAHON.** ParadisEO : Une plate-forme pour la conception et le déploiement de métaheuristiques parallèles hybrides sur clusters et grilles. Thèse de doctorat, 2005.

[29]. **T. A. Feo, M. G.C.Resende.** Greedy randomized adaptive search procedures. Journal of Global Optimization. 6 :109ï 133, 1995.

et M.P. Vecchi. Optimization by Simulated Annealing. 983.

[31]. **E.H.L. Aarts, J.H.M. Korst.** Simulated Annealing and Boltzmann machines. Chichester, Wiley. 1989.

[32]. **P.J.M. van Laarhoven, E.H.L. Aarts.** Simulated Annealing. Theory and Applications. Dordrecht, Kluwer. 1988.

[33]. **F.Glover.** Tabu Search, part I. ORSA, Journal of Computing, 1 :190ï 206, 1989.

[34]. **F.Glover**. Tabu Search, part II. ORSA, Journal of Computing, 2 :4ï 32, 1989.

[35]. **F.Glover**. Tabu search : A tutorial. In Interfaces, volume 20, chapter 4, pages 74ï 94. 1990.

[36]. **F.Glover, and M.Laguna.** Tabu search. In editor C.R.reeves, editor, Modern Heuristics Techniques for Combinatorial Problems. pages 70ï 150. Blackwell Scientific Publications, London, 1993.

[37]. **X.Delorme.** Modélisation et résolution des Problèmes Liés à lôexploitation dôinfrastructures ferroviaires. thèse de Doctorat ; lôuniversité de Valenciennes et Hainaut-Cambrésis ;10 Décembre 2003.

[38]. **Michel.Salomon.** Technique d'optimisation " Les métaheuristiques". I.U.T. Belfort-Montbéliard.

[39]. **M.O'Keeffe, et M.Ô Cinnéide.** A Stochastic Approach to automated Design Improvement. 2003.

[40]. **M.Soriano, et P.et Gendreau.** Fondements et application des méthodes de recherche avec tabous. RAIRO 31(2): 133-159. 1997.

[41]. **E.H.L.Aarts.** Simulated Annealing and Boltzmann machines. John Wiley & Sohns Ltd, 1989.

[42]. **M.G.A.VERHOEVEN, E.H.L. AARTS.** Parallel local search. Journal ofHeuristics 1(1) : 43-65, 1995.

[43]. **V.-D.Cung, S. L. Martins,C. C Ribeiro et C.Roucairol.** Strategies for the parallel implementation of metaheuristics. Essays and Surveys in Metaheuristics. C. C.R. a. P. Hansen, Kluwer: 263-308.2001.

[44]. **E.Cantû-Paz.** A survey of parallel genetic algorithms. Calculateurs parallèles,réseaux et systèmes répartis 10(2): 141-171.1998.

[45]. **M.R.GAREY, D.S.JOHNSON.** Computers and Intractability: A Guide to the Theory of NPCompleteness. W.H. Freeman, San Francisco 1979.

T. Lower Bounds for the Quadratic Assignment Problem : Operations Research 46, 912-922.1998.

[47]. **S.E.Karisch, R.E. Burkard and F. Rendl.** Qaplib - a quadratic assignment problem library.

[48]. **Hahn, et al.** The Quadratic Three-dimensional Assignment Problem: Exact and Approximate Solution Methods. 11-08-2005.