République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Mémoire de Magister

Présenté au centre universitaire Khenchela
Faculté des sciences et sciences de l'ingénierie

Département de : **Informatique**
Ecole **D**octorale **N**ationale **S**cience et **T**echnologie de l'**I**nformation et de la **C**onnaissance
Spécialité : **Informatique**
Option : **SIC**

Présenté par : **Mlle**: **Guefrouchi Ryma**

Métaheuristiques parallèles hybrides pour résoudre le problème Q3AP sur grille de calcul

JURY

Président	**Dr.**	**Djamel Eddine Saidouni**	**Université de Constantine**
Rapporteur	**Dr.**	**Kholladi M-Khier-Eddine**	**Université de Constantine**
Examinateur	**Dr.**	**Salim Chikhi**	**Université de Constantine**
Examinateur	**Dr.**	**Allaoua Chaoui**	**Université de Constantine**

2009

بسم الله الرحمان الرحيم

A ma mère

A mon père

A mon frère

A mes sœurs

A ma grande famille

A tous mes collègues et amis

Remercîments

Mon premier et dernier remercîment c'est au dieu, qui m'a donné le courage, la paissance, et la volonté pour pouvoir terminer ce travail,

ALHAMDO-LI- ALLAH.

Je remercie vivement docteur Kholladi Mohammed Khier-Eddine, mon directeur de mémoire, pour m'avoir dirigé pendant cette période de recherche. Les conseils, la confiance et la liberté qu'il m'a accordés au cours de ce travail, m'ont permis d'entreprendre de nombreuses expériences qui ont grandement contribué à la réalisation de ce mémoire.

J'aimerais bien remercie mes parents qui m'ont toujours aidé, soutenu, encouragé au cours de mes études.

Je réserve une reconnaissance à ma tente Meriem et ces filles qui n'ont pas cessé de m'encourager durant plus d'une année que j'ai passé à Guelma.

Je réserve un remercîment chaleureux à mes deux copines Zahra et Souad.

Mes derniers remercîments s'adressent à ma grande famille, à mes sœurs, mon frère, mes tentes, mes oncles, et tous qui m'ont encouragé et soutenu durant mes études.

Résumé

L'Affectation Quadratique, QAP, est un problème classique d'Optimisation Combinatoire, où on vise à chercher l'affectation optimale de n usines sur n sites, en minimisant le coût quadratique influencés par la distance inter-sites et le flux de matière inter-usines. Un nouveau problème constituant une extension du QAP, il s'agit de l'affectation quadratique en trois dimensions Q3AP, cette fois on cherche à optimaliser l'affectation de n usines et n managers sur n sites. Le coût quadratique dans ce cas dépend de la distance inter-sites, les volumes de matières échangés inter-usines et aussi du taux d'échanges inter-mangers. Peux de travaux dans la littérature ont été proposés pour résoudre ce problème, surtout pour les applications de grande de taille où l'utilisation des métaheuristiques est recommandée. L'approche des métaheuristiques dite « A solution unique », en se basant sur l'aspect de l'intensification, donne rapidement une bonne solution, mais risque d'être piégée par un optimum local. Ce ci est évité avec l'aspect de la diversification fourni par l'approche « à population de solution », cette dernière possède le désavantage que l'ensemble de solutions trouvées est juste une approximation de l'optimum .Ce pendant on voit que l'hybridation des deux approches peut mieux guider le processus de la recherche vers la solution optimale, le mécanisme du parallélisme basé sur l'utilisation des grille de calcul, constitue une puissance de calcul et peut accélérer le processus de l'optimisation. Afin de saisir les gains de cette hybridation parallèle on propose dans ce mémoire une application de cette dernière sur les deux problèmes QAP et Q3AP, notre expérimentation est réaliser sur la plateforme Paradiseo dédiée à la conception des métaheuristique parallèles hybrides.

Mots-clés : QAP, Q3AP, Métaheuristique, hybridation, parallélisme, grille de calcul.

ABSTRACT

The Quadratic Assignment, QAP is a classic problem of combinatorial optimization where the aim is to find the optimal assignment of n facilities to n locations, minimizing the quadratic influenced by the distance between sites and the international flows of matter factory. A new problem is an extension of the QAP, it is the quadratic assignment Q3AP in three dimensions, this time we seek to optimize the allocation of n and n plant managers on n sites. The quadratic in this case depends on the distance between sites, the volume of materials exchanged inter-plant and also the rate of inter-mangers. Few works in the literature have been proposed to solve this problem, especially for applications of large size where the use of metaheuristics is recommended. The approach of metaheuristics called "A single solution", based on the aspect of intensification, gives a good solution quickly, but may be trapped by a local optimum. This is avoided with the aspect of diversification provided by the approach to solution of population ", the latter has the disadvantage that the set of solutions found is just an approximation to the optimum. This shows that during the hybridization of the two approaches can best guide the research process towards the optimal solution, the mechanism of parallelism based on the use of grid computing, provides computing power and can accelerate the process of optimization. To capture the gains of the parallel hybridization proposed in this paper an application of the latter on both issues QAP and Q3AP, our experiment is performed on the platform Paradiseo dedicated to the design of parallel hybrid metaheuristic.

Keywords: QAP, Q3AP, metaheuristics, hybridization, parallelism, grid computing.

الملخص

QAP إحالة من الدرجة الثانية، مشكلة كلاسيكية من التحسين التوافقي ، حيث الهدف هو العثور على الإحالة الأمثل ل«ن » مصنع في« ن» مواقع ، التقليل من تكلفة الإحالة يتأثر المسافة بين مواقع و قيمة التدفقات بين المصانع. وهناك مشكلة جديدة هي امتداد للQAP ، هي Q3AP إحالة في ثلاثة أبعاد ، وهذه المرة نحن نسعى إلى تحسين توزيع«ن» مصنع و « ن» مسير على « ن» مواقع. في هذه الحالة يعتمد على المسافة بين المواقع، وحجم المواد المتبادلة بين المصانع وكذلك معدل تبادل المعلومات بين مديريها. عدد الأعمال و الاقترحات لحل هذه المشكلة قليل، لا سيما بالنسبة للتطبيقات ذات حجم كبير حيث يوصى باستخدام metaheuristics. نهج metaheuristics المعتمد على "حل واحد"، يستند إلى تكثيف البحث حول هذا الحل، ويعطي حلا جيدا بسرعة ، ولكن قد يكون محاصرا من قبل الأمثلة المحلية. هذا ما يتجنبه نهج "حل المجموعات"، وهذه الأخيرة لها مساوئ أن مجموعة من الحلول هي مجرد حل تقريبي إلى الأمثل ، و من هنا استنتجنا أن التهجين بين النهجين يمكن أن يقدم حل أفضل بعد عملية البحث عن الحل الأمثل ، وآلية التوازي على أساس استخدام الحوسبة الشبكية ، يوفر القدرة الحاسوبية ويمكن الإسراع في عملية التحسين. للتأكد من هذه المكاسب نقترح في هذه الورقة تطبيق هذا الأخير على القضايا QAP Q3AP ، تجربتنا يتم تنفيذها على Paradiseo قاعدة مخصصة لتصميم metaheuristic .

الكلمات الرئيسية : QAP، Q3AP، metaheuristics ، التهجين ، التوازي ، الحوسبة الشبكية

TABLE DES MATIERES

LISTE DE FIGURES

LISTE DES TABLEAUX

Introduction

Le Problème d'optimisation combinatoire PMO est imposé dans une large gamme des domaines applicatifs, tant en Informatique, en Production, en Télécommunication, en Bioinformatique et autres. Il consiste, pour un ensemble de variables prenant des valeurs entières, à déterminer leurs instanciations, éventuellement sous contraintes, correspondant au maximum (ou minimum) d'une ou plusieurs fonction de ces variables, généralement appelées fonctions objectives OF [1]. L'ensemble des valeurs pouvant être prises par ces variables, constitue l'espace de recherche du problème.

La modélisation du problème constitue une étape importante avant de parler de la résolution de celui ci, elle se résume en plusieurs points importants nécessitants une bonne connaissance du domaine d'application:

- la définition de l'espace de recherche du problème : c'est à dire l'ensemble de solutions réalisables.
- la formulation des objectifs à optimiser : l'optimisation peut être mono-objective si on a une seule fonction à optimiser, sinon elle est multi-objective.
- la formulation des contraintes à prendre en considération.

Classification des méthodes de résolution des PMO

L'importance des problèmes d'optimisation combinatoire, et leurs larges domaines d'application a donné naissance à des nombreuses méthodes de résolution, en recherche opérationnelle (RO) et en intelligence artificielle (IA). Les méthodes exactes tel que le "branch and bound", l'algorithme A*, et la programmation dynamique sont efficaces pour des problèmes de petites tailles [3], Elles garantissent l'optimalité de la solution, mais elles sont complexes avec un temps important de la résolution [4]. Pour des problèmes de grandes tailles, les procédures exactes ont montré leurs limites. On a fait recours aux méthodes heuristiques pour résoudre ce type de problèmes. Les algorithmes heuristiques tels que le recuit simulé, la recherche tabous et les algorithmes génétiques essaient de trouver rapidement de bonnes solutions, qui ne sont pas forcément des solutions optimales [5]. À leur tour les heuristiques peuvent être divisés en deux classes : d'une part les algorithmes spécifiques à un problème donné connues par les heuristiques spécifiques et qui nécessitent des connaissances du domaine [6], et d'autre part les algorithmes généraux

applicables à une grande variété de problèmes connus par les métaheuristiques au quelle on s'intéresse.

La notion des métaheuristiques parallèles hybrides est récente, elle est apparue après le grand pas qu'ont fait les technologies informatiques, que ce soit matériel ou logiciel. Les deux concepts : parallélisme et hybridation sont utilisés dans le but d'améliorer les performances des algorithmes d'optimisation et les accélérer surtout quand il s'agit d'un problème de grande taille, où les processus de résolution deviennent bloquants, la distribution des calculs et des donnés constitues l'idée de plusieurs projet de recherches. Dans les mêmes buts, la technologie des grilles de calcul et le concept de la gridification sont introduits.

En contribuant dans l'optimisation combinatoire, nous avons proposé d'utiliser un modèle des métaheuristiques parallèles hybrides, où un algorithme génétique coopère avec un algorithme Hill Climbing pour résoudre un type particulier des PMO, il s'agit de l'affectation quadratique, et l'affectation quadratique en trois dimensions, deux problèmes d'optimisation combinatoire mono-objective. Pour pouvoir mesurer les performances des modèles proposés, on a implémenté ces modèles sous la plateforme Paradiseo dédiées à la conception et implémentation des métaheuristiques parallèles hybrides.

Notre mémoire est divisé en quatre chapitres, le premier est intitulé Métaheuristiques Parallèles Hybrides Pour L'optimisation Combinatoire Sur Grille De Calcul, il constitue un état de l'art de ce domaine. Le deuxième chapitre est une présentation des deux problèmes d'optimisation combinatoire : QAP et Q3AP. Dans le troisième chapitre et à travers le premier, on proposera une solution des problèmes présentés précédemment. Dans le quatrième chapitre on présente notre implémentation, ainsi que les résultats obtenus. On terminera notre mémoire par une conclusion et des perspectives.

CHAPITRE 1
METAHEURISTIQUES PARALLELES HYBRIDES POUR l'OPTIMISATION COMBINATOIRE SUR GRILLE DE CALCUL

1. Introduction

Ce chapitre est divisé en quatre sections, dans la première section on présentera les métaheuristiques et leurs algorithmes, que ceux soient à solution unique ou à population de solutions, et en particulier on détaillera les algorithmes génétiques. Dans la deuxième section on mettra l'accent sur les différents modèles parallèles des métaheuristiques, ainsi que les possibilités d'hybridation entre métaheuristiques. Dans la troisième section notre intérêt sera la notion des grilles de calcul et particulièrement notre cadre d'utilisation de cette technologie. Dans la dernière section on présentera la plateforme Paradiseo dédiée à la conception des métaheuristiques parallèles hybrides ainsi que les possibilités du déploiement des métaheuristiques parallèles sur grille de calcul.

2. Les métaheuristiques pour l'optimisation combinatoire

Apparues dans les années 1980, les métaheuristiques, forment un ensemble d'algorithmes, visant à résoudre une large gamme de problèmes d'optimisation difficiles, pour lesquels on ne connaît pas de méthodes classiques plus efficaces [7]. Les métaheuristiques sont des méthodes itératives, progressives, commençant par comprendre les caractéristiques d'un problème donné avant de proposer un processus de résolution [8].

2.1.Approches pour les métaheuristiques

Deux approches des métaheuristiques sont possibles [9]:

À solution unique : consiste à évaluer une seule solution à chaque itération, cette approche est fondée sur la notion de parcours du voisinage de la solution courante afin d'intensifier la recherche dans les régions prometteuses (Figure 1.1). Parmi les algorithmes les plus connus de cette approche, on cite le recuit simulé, la recherche Tabou et la descente du gradient.

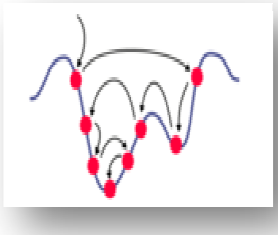


Figure 1.1 Métaheuristiques à solution unique.

À population de solutions : elles consistent à évaluer un ensemble de solutions à chaque itération (Figure 1.2). On vise à visiter des bonnes solutions bien réparties dans l'espace de la recherche. Il s'agit de la diversification et l'exploration de cet espace. Les algorithmes évolutionnaires constituent une partie importante de cette catégorie de méthodes.

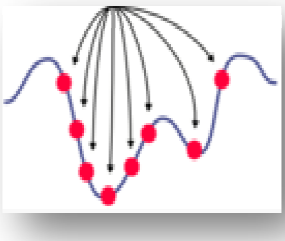


Figure 1.2. Métaheuristiques à population de solutions.

2.2.Taxonomie pour les métaheuristiques

On peut schématiser la taxonomie des métaheuristiques de la manière suivante (Figure 1.3) [9], [10], [22]:

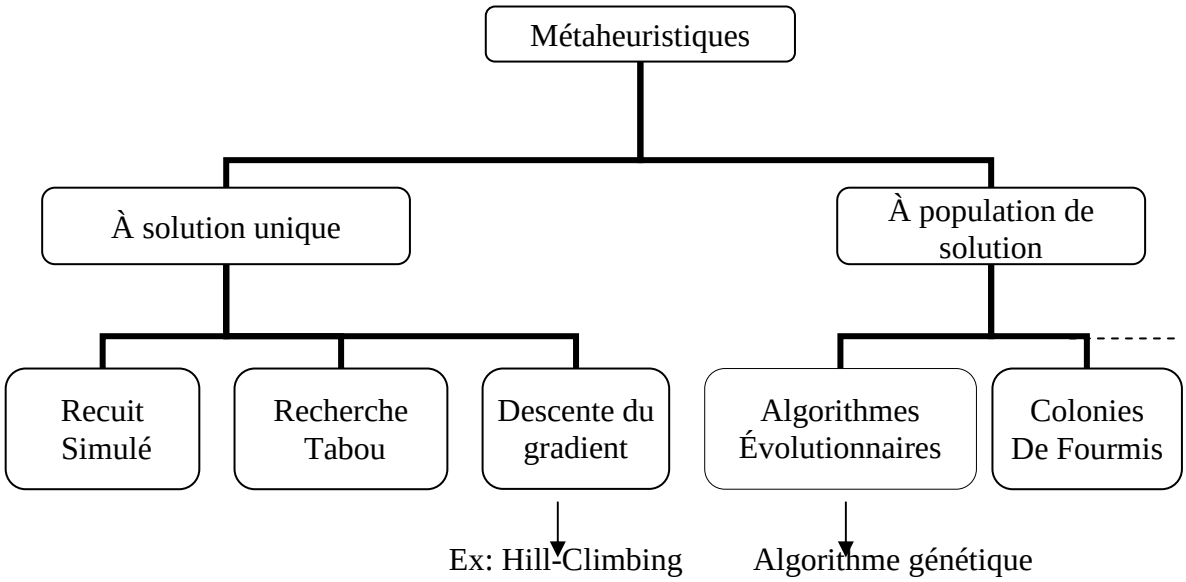


Figure 1.3. Taxinomie des métaheuristiques.

2.3.Algorithmes pour les métaheuristiques à solution unique

D'abord il faut donner deux définitions nécessaires à la compréhension du fonctionnement de ce type d'algorithmes [21] :

Un mouvement : est une opération élémentaire permettant de passer d'une solution à une solution voisine (exemple : changer la valeur d'une variable, échanger deux variables).

Le voisinage d'une solution : est l'ensemble des solutions voisines, c'est à dire l'ensemble des solutions accessibles par un mouvement (et un seul).

Donc le mouvement est un opérateur élémentaire appliqué pour transformer une solution courante à une nouvelle solution voisine. L'ensemble des solutions pouvant être générées à travers un mouvement composent un voisinage.Maintenant, on citera quelques algorithmes à solution unique, pour voir leurs principes de fonctionnement et comment il progresse pour trouver la solution optimale.

2.3.1. Le recuit simulé

Cet algorithme s'est inspiré du phénomène physique du recuit, appliqué sur les matériaux solides. L'algorithme de recuit simulé suit les étapes suivantes [11], [12], [13]:

- Partant d'une solution initiale choisie aléatoirement, avec une température initiale élevée, on prend une solution voisine puis on évalue le coût.
- Si le coût de la solution voisine est inférieur à la solution courante, on prend cette solution comme nouvelle solution courante.
- Sinon on prend cette solution voisine comme nouvelle solution courante avec probabilité proportionnelle à la température. Ainsi on évite d'être bloqué par un optimum local.
- On diminue la température avec un facteur appartenant à l'intervalle [0,1]
- Relancer ces étapes jusqu'à un critère d'arrêt.

2.3.2. La recherche Tabou

Cet algorithme est mis en œuvre pour remédier au problème de l'algorithme précèdent, deux modification sont apportées [11], [12], [13]:

1- la recherche guidée : la recherche doit être efficace et ne s'appuie pas sur le hasard.
2- La recherche intelligente : en utilisant une mémoire pour sauvegarder les points déjà visitées, évitant ainsi de les visiter encore une fois dans le futur.

2.3.3. La descente du gradient

Appelé aussi Hill Climbing, une méthode très simple, consiste à [11], [12], [13], [14] :

- Prendre dans un premier temps une solution initial s_0 possédant une fonction objective $F(s_0)$.
- On génère $V(s_0)$, l'ensemble de solutions voisines de s_0.
- Dans un deuxième temps, on va chercher une solution s appartenant à $V(s_0)$, qui minimise tout les $F(s)$, étant **s** appartient à $V(s_0)$.
- Si $F(s)$ est inférieur à $F(s_0)$, il faut réitérer ces étapes avec s.
- Sinon on va prendre s_0 comme solution optimale.

Le premier inconvénient remarqué dans cette méthode est le fait qu'il faut visiter tous les points s et calculer leurs fonctions objectif afin de trouver l'optimum. Dans le cas des problèmes de grande taille, le temps de la recherche devient inacceptable. L'existence de minima locaux et la possibilité d'être piégé par ces minima, constitue un deuxième handicape de cette méthode (Figure 1.4).

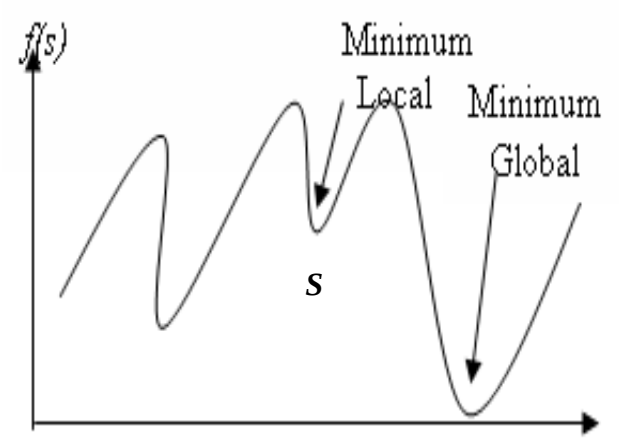


Figure 1.4. Définition du minimum local/ global.

2.4.Algorithmes pour les métaheuristiques à population de solutions

Maintenant notre intérêt est la deuxième approche des métaheuristiques, dite à population de solutions, quelques algorithmes sont cités ainsi que leurs principes de recherche de la solution optimale.

2.4.1. L'algorithme de colonies de fourmis

Cet un algorithme itératif à population de solutions, il est inspiré du comportement des fourmis, qui disposent de capacités très limitées et à travers la coopération elles peuvent résoudre des problèmes très difficiles. En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on conclut qu'elle cherche à résoudre un problème du plus court chemin [15]. Les individus partagent une mémoire commune de connaissance, contenant les solutions déjà visitées par les individus : à chaque fois qu'un individu découvre une solution au problème : les meilleures ainsi que les mauvaises, il enrichit la mémoire de connaissance collective de la colonie, et à chaque fois qu'un individu fait un choix il s'appui sur cette mémoire de connaissance collective [16], [17].

2.4.2. Les algorithmes évolutionnaires

Les algorithmes génétiques (AG) constituent une grande partie de cette catégorie d'algorithme. Le principe d'un AG est simple, il est inspiré de la théorie de l'évolution proposée par C. Darwin [18]. Il est motivé par sa capacité d'adaptation au problème de différentes natures. Un algorithme génétique manipule le codage de l'ensemble des paramètres du problème à résoudre et non pas les paramètres eux même [19], donc le codage de l'ensemble des paramètres du problème d'optimisation en des chaînes de longueurs finies dites chromosomes constitue une étape préalable d'un AG. Ce dernier est un processus itératif : on part avec une population initiale d'individus (solutions)

aléatoirement générées, auxquels on évalue une fonction d'adaptation (la fonction objective), puis à chaque itération, les individus les mieux adaptés sont sélectionnés (en se basant sur la fonction d'adaptation) pour participer à l'opération de croisement et de mutation. De nouveaux individus seront générés, ces derniers seront introduits à la place des individus les moins adaptés [20], comme ça la nouvelle population est générée. Ce processus est itéré jusqu'à un critère d'arrêt. Le critère, le plus couramment utilisé, est le nombre maximal de générations que l'on veut effectuer [18]. À l'aide de ces mécanismes de reproduction (Mutation et Croisement), cet algorithme permet l'exploration de l'espace de solutions et de fournir des points éloignés à ceux déjà visités (Figure 1.5).

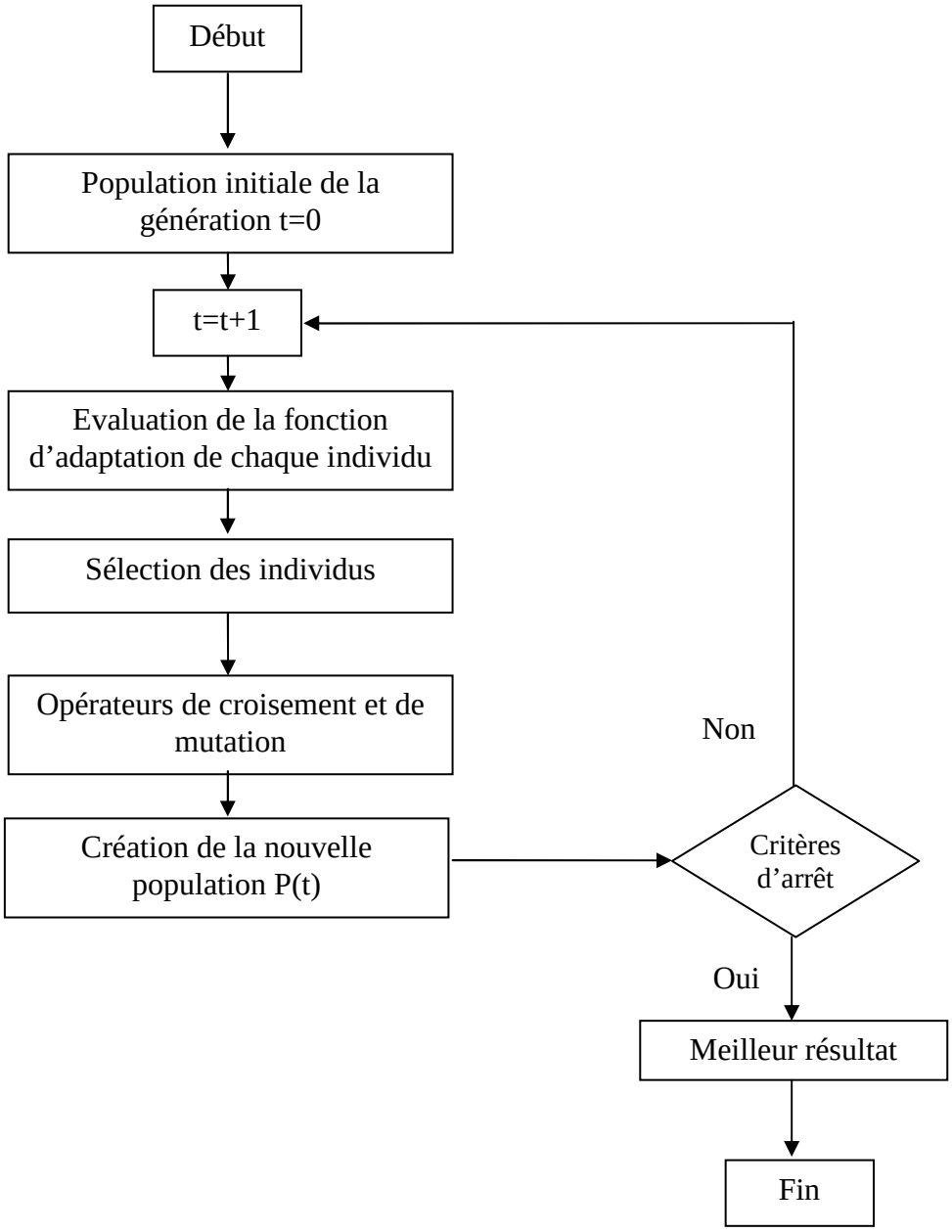


Figure 1.5. Organigramme des algorithmes évolutionnaires [18].

a. Le codage des individus

Le codage utilisé par un AG est représenté sous forme d'une chaîne finie qui contient toute l'information nécessaire pour représenter un point de l'espace de recherche. Les algorithmes génétiques sont appliqués sur une population d'individus ou de chromosomes. Un chromosome est composé de gènes, chaque gène représente un paramètre ou une information sur l'individu, chaque individu est représenté par un ensemble de chromosomes. Une population est un ensemble d'individus. Le codage binaire est le code le plus utilisé. Mais il existe d'autres types de codage tels que le codage de Gray et le codage réel [23].

- Pour le codage binaire par exemple, les gènes sont représentés par des bits et les chromosomes par des chaînes de bits [23].
- Pour le codage réel, les gènes sont représentés par des valeurs réelles, il est utile quand on cherche à optimiser une fonction réelle [23].
- Le codage de Gray est un codage qui garantit la propriété : « il y a une différence d'un seul bit entre un élément n et son voisin n + 1 dans l'espace de recherche ».

  Donc la "distance de Hamming" utilisée fréquemment comme mesure de la dissimilarité sera efficace, cette mesure compte les différences de bits de même rang de deux séquences de bits, deux éléments voisins en terme de distance de Hamming codent deux éléments proches dans l'espace de recherche. Cette propriété n'est pas offerte par le codage binaire [7], [23].

b. La fonction fitness (fonction d'évaluation)

Elle constitue une étape indispensable pour la phase de sélection dans les algorithmes génétiques. Chaque chromosome dans l'espace de recherche est soumis à une procédure de calcul de sa performance. A partir de cette mesure, on peut différencier entre les chromosomes par l'efficacité de chacun par apport aux autres [24].

c. Opérateur de sélection

Cet opérateur est chargé de définir quels seront les individus de la population initiale qui vont être sélectionnés et groupés deux à deux pour participer à l'opérateur de recombinaison. La taille de la population ne varie pas après l'exécution de l'algorithme génétiques, on doit donc sélectionner *n/2* individus parmi *n* individus de la population initiale (l'opérateur de croisement nous permet de repasser à *n* individus). Cet opérateur est le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de sélection d'un individu est

liée à son adaptation ou sa fitness. On va discuter trois types de méthodes de sélection différentes [24] :

La méthode de la "loterie biaisée" (roulette wheel) de GoldBerg

Avec cette méthode chaque individu a une chance d'être sélectionné proportionnellement à sa performance, on utilise l'image de la "roue du forain", à chaque individu on attribue un angle relatif à son adaptation ou sa fitness. Donc plus les individus sont adaptés au problème, plus leurs angles sont grands et plus ils ont de chances d'être sélectionnés. On fait tourner la roue et quand elle cesse de tourner on sélectionne l'individu correspondant au secteur désigné par un "curseur". Cette méthode a pas mal d'inconvénients : il n'est pas impossible que sur toutes les sélections visant à désigner les parents de la nouvelle génération, les individus sélectionnés soient des mauvais individus. Ce phénomène ne marche pas avec le principe des algorithmes génétiques qui dit que les meilleurs individus soient sélectionnés pour converger vers une solution la plus optimale possible. A l'inverse, on peut avoir un individu "localement supérieur" c'est-à-dire ayant une fitness trop élevée par apport aux autres individus de la population initiale, il sera presque toujours sélectionné avec un angle qui constitue la quasi-totalité de la roue. Après quelques générations successives, on se retrouve avec une population ne contenant que des copies de cet individu, alors qu'il n'est pas globalement supérieur.

La sélection élitiste

Cette méthode consiste à trier les individus de la population initiale, de manière décroissante selon leur fitness et sélectionner parmi eux les meilleurs individus. Il est claire que cette méthode mènera vers une convergence prématurée et encore très rapidement et sûre.

La sélection par tournois

Cette méthode consiste à effectuer un tirage avec remise de deux individus de la population initiale, celui qui est le plus adapté, possédant la fitness la plus élevée lest sélectionné avec une probabilité p comprise entre 0.5 et 1. Ce processus est répété autant de fois que le nombre d'individus qu'on veut sélectionner. On remarque qu'on peut maintenant guider la sélection par la modification de la valeur p.

d. L'opérateur de croisement

Cet opérateur est appliqué après l'opérateur de sélection sur la population initiale de taille *n.* Il permet de créer de nouveaux individus appelée enfants, les *n/2* individus sélectionnés seront couplés deux à deux pour former n/4 couples qui vont donner naissance à n/4 nouveaux individus, ainsi on repasse à n individus comme on a démarré en échangeant de l'information entre deux chromosomes ou individus appelés parents. La recombinaison la plus utilisée est celle qui recombine deux individus au niveau d'un seul locus (position) (Figure 1.6) [19].

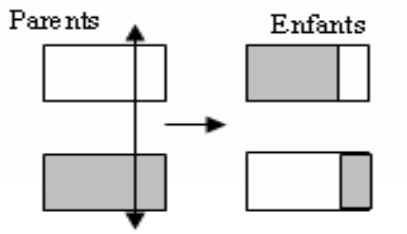


Figure 1.6. Croisement sur une seule position.

Il est possible de recombiner deux individus au niveau de deux locus ou plus comme il est montré dans la Figure 1.7 :

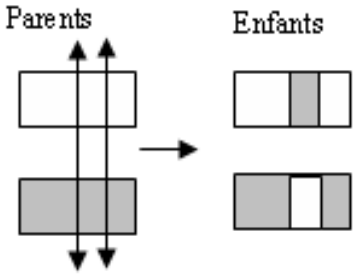


Figure 1.7. Croisement sur deux positions.

Le nombre de points de croisements et leurs positions sont tirés au hasard, l'opérateur est appliqué avec une probabilité de croisement. Cet opérateur introduit une grande diversité du fait que plus le nombre de points de croisements sera grand et plus la probabilité de croisement sera élevée plus la diversité sera aussi grande. Néanmoins, il est possible de rencontrer le problème suivant, la solution optimale possède un gène à une position particulière, avec une valeur bien précise, alors que aucun individu de la population initiale ayant cette valeur à cette position, ainsi la solution optimale ne sera jamais aboutie à travers les opérateurs de sélections et de recombinaison.

e. L'opérateur de mutation

Cet opérateur permet d'engendrer un nouvel individu à partir d'un opérateur parent en modifiant un gène aléatoirement choisi. Pour un codage binaire cela revient à changer un 1 en 0 et vice-versa (Figure 1.8). Cet opérateur introduit de la diversité dans le processus de recherche des solutions et peut aider l'AG à ne pas stagner dans un optimum local. A travers cet opérateur, il sera possible d'atteindre tous les points de l'espace de recherche permettant ainsi de remédier au problème de l'opérateur de recombinaison décrit précédemment.

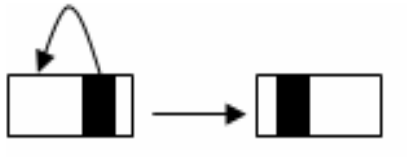


Figure 1.8. L'opérateur de mutation.

f. L'opérateur de remplacement

Cet opérateur est le plus simple, son travail consiste à réintroduire les descendants obtenus par application successive des opérateurs de sélection, de croisement et de mutation dans la population initiale. On trouve deux méthodes de remplacement différentes [19], [23],[24]:

Le remplacement stationnaire :

Dans ce cas, les enfants remplacent automatiquement les parents, même si un enfant possède une fitness très basse. Il remplace un parent qui peut être le plus adaptés, deux stratégies existent :

- Remplacer la totalité de la population initiale par la nouvelle population.
- Choisir une certaine proportion d'individus de la nouvelle population pour remplacer leurs parents.

Le remplacement élitiste :

Dans ce cas, on garde au moins l'individu le plus adaptés d'une génération à la suivante, on peut partir du principe qu'un nouvel individu (enfant) prend place au sein de la population que s'il remplit le critère d'être plus performant que le moins performant des individus de la population précédente.

3. Métaheuristique parallèles hybrides

La complexité des problèmes d'optimisation et leurs larges domaines d'applications ainsi que les limites des différentes méthodes standards de résolution notamment quand il s'agit d'un problème de grande taille, ont poussé les recherches vers nouvelles techniques d'amélioration de qualité de solution ainsi que le temps d'obtention de résultats. Cette section s'articule sur deux mécanismes : le parallélisme et l'hybridation des métaheuristiques pour les deux approches des métaheuristiques : à population de solution et à solution unique, beaucoup des modèles parallèles et hybrides ont été proposé pour résoudre les problèmes d'optimisation NP-difficiles pour lesquels il n'existe pas d'algorithmes produisant une solution optimale en un temps d'exécution acceptable par rapport à la taille des données manipulées. Ces mécanismes ont montré leur efficacité de résolution pour beaucoup de problèmes [21]:

- Les voyageurs de commerce ;
- Le problème de l'affectation quadratique ;
- Le problème sac à dos ;
- Et le problème de routage des véhicules.

Dans cette section on va motiver les mécanismes de parallélisme et hybridation, dans la première sous section on s'intéresse aux modèles parallèles des métaheuristiques, on va prendre les algorithmes génétiques comme exemple des métaheuristiques à population de solutions et les algorithmes de recherche locale pour les métaheuristiques à solution unique, (le même principe pour les autres algorithmes de même type). Dans la deuxième sous section notre intérêt sera l'hybridation des métaheuristiques.

3.1.Parallélisme des métaheuristiques

Le parallélisme des métaheuristiques est motivé par la distribution des calculs et le déploiement de l'exécution des différents modèles parallèles. La Figure 1.9 présente une taxinomie hiérarchisée des différentes formes de parallélisassions et de coopération propres aux métaheuristiques [21], [09].

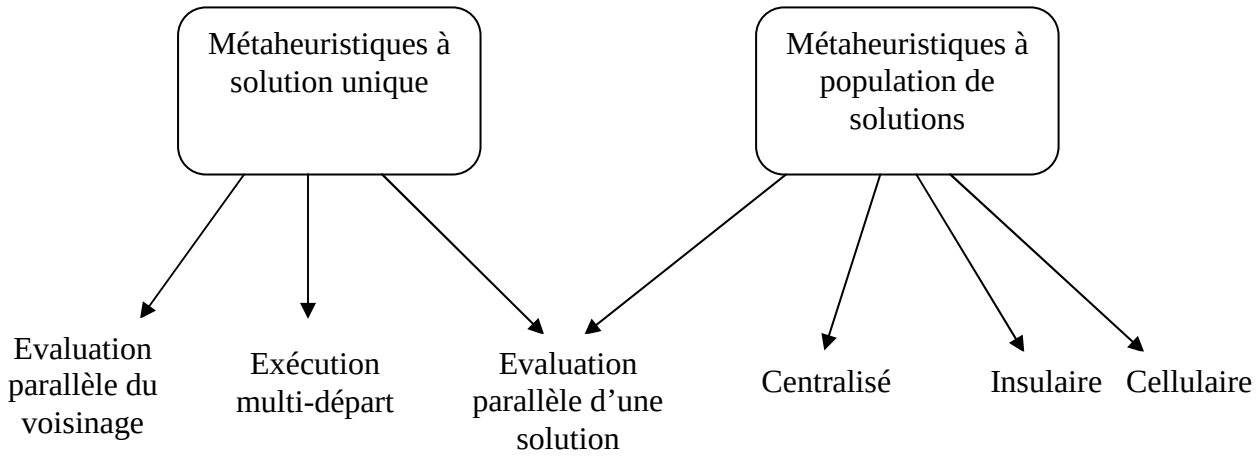


Figure 1.9. Coopèration des métaheuristique.

3.2.Modèles parallèles pour les métaheuristiques à population de solution

Le parallélisme des métaheuristiques à population de solution est motivée par le gain du temps d'obtention des résultats par une distribution de calculs et de recherches, en plus le déploiement de l'exécution des modèles parallèles basés sur l'évolutions concurrentes et coopératives de populations distribuées constitue une puissance de ces modèles.

3.2.1. Le modèle insulaire coopératif

Ce modèle est inspiré du comportement des niches écologiques. La population initiale est divisée en sous population ou île, évoluant simultanément et coopérant à travers le flux de migrations de solutions, périodiques ou non, entre les îles connectées (Figure. 1.10).

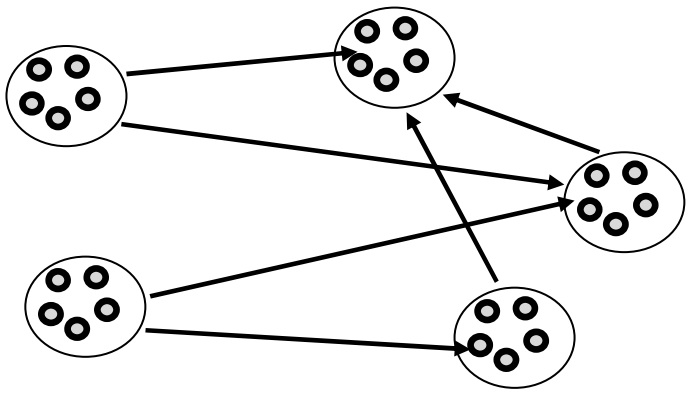


Figure 1.10. Le modèle insulaire coopératif.

Sur chacune des sous-populations, le processus d'évolution s'effectue sur un normalement, commençant par évaluation des individus, suivie par sélection des individus les mieux adaptés pour participer aux opérations de variation (recombinaison et mutation), l'exécution de ces derniers donne naissance à des nouveaux individus qui vont remplacer d'autres individus moins adaptés dans la sous population, offrant ainsi une diversification et exploration de l'espace de recherche. La gestion des migrations intervient à chaque génération de l'AG. standard, succédant à la phase de remplacement. On distingue les modèles synchrone et asynchrone. Il est nécessaire de définir plusieurs paramètres de gestion des migrations:

- La topologie du modèle coopératif ;
- Le critère de décision d'immigration ;
- Les individus qui seront immigrés ;
- Et les individus qui seront remplacés par ceux immigrés.

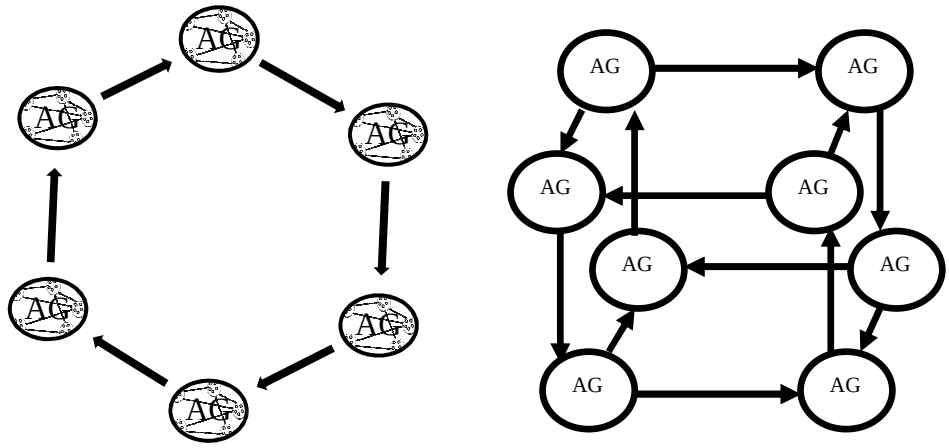
Une mémoire est utilisée dans ce modèle coopératif, à un instant donné elle est constituée des individus en cours de migration d'une sous population vers une autre, rendant ainsi ce modèle tolérant à la perte des individus immigrés.

Dans une implémentation synchrone, le processus de migration intervient de manière périodique. Après un nombre constant d'itérations de l'AG chaque îles émet des solutions vers ses populations voisines, et attend de leur part de nouvelles solutions avant de poursuivre le processus d'évolution. Ce modèle est synchronisé par le nombre fixe d'itérations de l'AG sur tous les sous population. Remarquons que le temps d'exécution d'un même nombre d'itérations est différencié par le type d'opérateurs de variation et le support d'exécution qui peuvent être hétérogènes, on conclut que ce modèle est synchronisé par l'AG le plus lent.

Le modèle asynchrone associe un critère de décision d'immigration. En raison de l'hétérogénéité logicielle et matérielle, il se peut que les populations soient à différentes étapes d'évolution, pour cela le critère associé est vérifié à chaque itération, dans le cas positif des requêtes sont émises vers les populations voisines, et le processus d'évolution ne sera pas coupé, après un temps non déterminé de traitement les nouvelles solutions seront réceptionnées et intégrées dans la sous population.

Concernant la topologie d'interconnexion des îles, de nombreux travaux ont mesuré l'impact de leur configuration sur la qualité des résultats obtenus [23], [24]. Il apparait que les graphes cycliques sont préférables, les modèles en anneau et hyper cube (Figure. 1.11) sont d'ailleurs largement utilisés. Remarquons qu'un nombre trop élevé de jonctions se

révèle inefficace, l'ensemble des populations distribuées se comportant alors comme une seule population globale.



Topologie anneau Topologie hypercube

Figure 1.11. Topologies du modèle insulaire coopèratif.

3.2.2. Le parallélisme de la phase de transformation

Cette phase est la plus coûteuse en temps CPU, elle synthétise la recombinaison, la mutation et l'évaluation des nouveaux individus. Elle peut être modélisée selon le modèle « maître/esclave» synchrone ou asynchrone :

a. Le modèle parallèle synchrone :

A chaque itération, deux processus s'exécutent : «maître» et «esclave», le processus «maître» s'occupe de la gestion d'évolution de la population, en commençant par la sélection, passant par la transformation, et terminant par le remplacement. Il distribue l'ensemble des nouvelles solutions générées entre plusieurs agents évaluateurs, et il attend de leurs part les résultats d'évaluation, à la fin de collection des résultats le processus d'évolution se poursuit à nouveau.

Plusieurs inconvénients sont détectés pour ce modèle :

- Le processus d'évolution ne peut poursuivre avant le retour de la dernière solution évaluée.
- Il n'est pas tolérant aux pannes, la destruction d'un processus évaluateur nécessite une redistribution des solutions il est indispensable donc d'utiliser une mémoire pour sauvegarder les individus non encore évalués.

b. Le modèle parallèle asynchrone « Steady-State » :

Offre deux avantages en comparaison du modèle précèdent. Il est naturellement tolérant aux pannes, il n'y a en effet de contrôle quant au retour des solutions émises pour évaluation. De plus, ce modèle est bien adapté, en termes d'efficacité, à l'ensemble des applications caractérisées par un cout à l'évaluation irrégulier. Il n'est pas nécessaire de mémoriser les solutions traitées par les processus évaluateurs. Enfin, l'extensibilité de ce modèle à l'exécution parallèle n'est pas limitée.

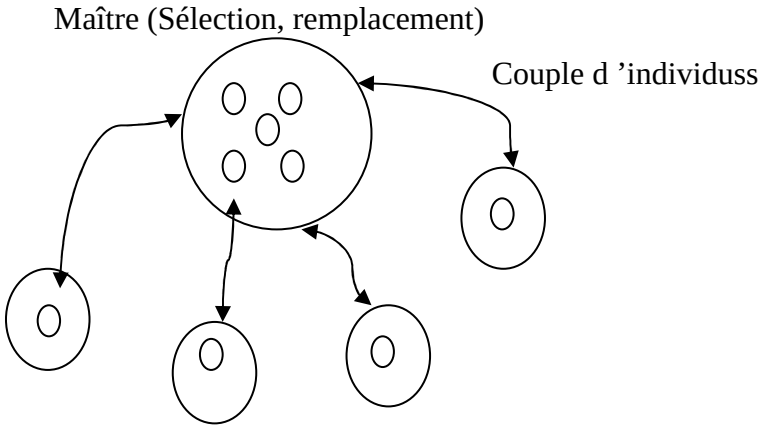


Figure 1.12. Modèle parallèle asynchrone pour la phase de transformation/ évaluation.

3.2.3. Le parallélisme de l'évaluation de la fonction objectif

Lorsque le coût de l'évaluation d'une seule solution devient important, ou quand elle nécessite l'accès à des bases de données volumineuses, l'utilisation d'un modèle parallèle pour cette phase (évaluation de la fonction objectif) est recommandée. Evaluer une seule solution d'une manière parallèle en utilisant le modèle parallèle maître esclave, consiste à répartir et émettre cette solution vers différents agents évaluateurs qui effectuent des calculs partiels, ces calculs seront collectés par la suite par l'algorithme « Maître ». Ce modèle est toujours synchronisé avec le retour de tous les calculs partiels, comme il n'nécessite la sauvegarde de ces calculs (Figure 1.12) [09].

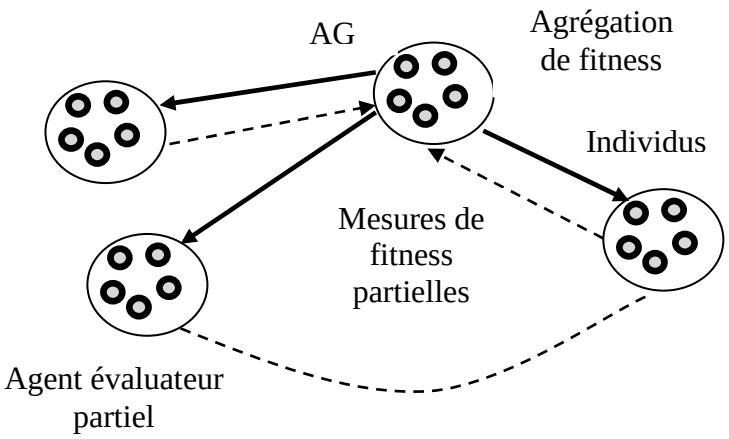


Figure 1.13. Parallèlisme de l'évaluation de la fonction objectiv.

3.2.4. Le modèle cellulaire

Dans ce modèle, la population est composée d'un ensemble de cellules, chacune d'elles est composée d'un individu et de son voisinage sur une structure topologique, généralement une grille torique. Chaque individu est semi-isolé, et n'a de visibilité que vers les solutions de sa cellule. Les phases de sélection, recombinaison et remplacement sont ainsi appliquées simultanément dans chaque cellule. En raison d'une convergence plus lente par rapport à l'A.E standard, ce modèle est étudié même dans les architectures séquentielles (Figure 1.13) [09].

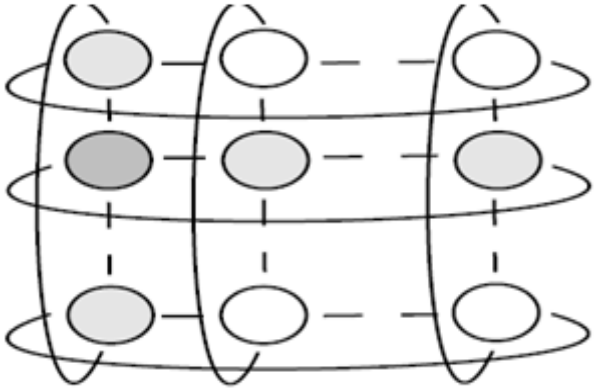


Figure 1.14. Le modèle cellulaire sur une grille torique.

3.3.Modèle Parallèle pour les métaheuristiques à solution unique

Trois modèles sont généralement utilisés dans la littérature : le modèle multi-départ, le modèle d'évaluation parallèle de voisinage et le modèle parallèle de mouvement.

3.3.1. Le modèle parallèle multi-départ

Consiste à lancer simultanément plusieurs algorithmes à base de recherche locale qui peuvent être [09] :

-Homogènes ou hétérogènes : en ce qui consterne la structure du voisinage et les opérateurs associés.

- Démarrant d'une même solution ou non : la solution de départ peut être la même pour tous les algorithmes en concurrence, il est aussi possible que chaque algorithme départ avec une solution différente aux autres.

-Indépendants ou coopératifs : les algorithmes indépendants n'échangent pas d'information durant leur exécution. Dans une forme coopérative du modèle multi-départ parallèle, des informations sont échangées entre les différentes méthodes à base de recherche locale. Ces informations peuvent être de bonnes solutions visitées, des mouvements parcourus, des paramètres exprimant la démarche suivie dans l'espace de recherche, etc.

Ce modèle parallèle est motivé par la facilité d'implémentation, et son indépendance du problème traité (Figure 1.14).

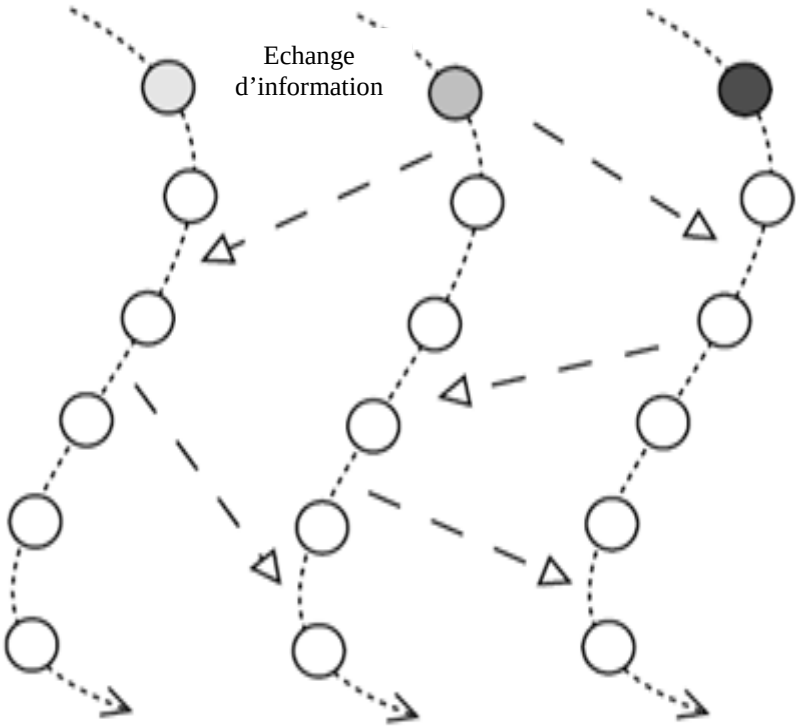


Figure 1.15. Le modèle coopèratif multidépart.

3.3.2. L'évaluation parallèle du voisinage

Lorsque le temps d'évaluation d'une solution voisine est important ou le nombre de solutions voisines d'une solution courantes est élevé. Le parallélisme de la phase d'évaluation parallèle du voisinage est recommandé, le modèle de type « Maître/Esclave » caractérisé par une distribution des calculs est utilisé dans ce contexte. À chaque itération, le processus « Maître » duplique et émet la solution courante vers tous les processus esclaves. Chaque processus esclave considère un voisinage partiel de la solution courante, les mécanismes de partitionnement du voisinage sont très variés. L'algorithme « Maître » se met en attente des résultats avant de poursuivre de nouveau (Figure 1.15) [09].

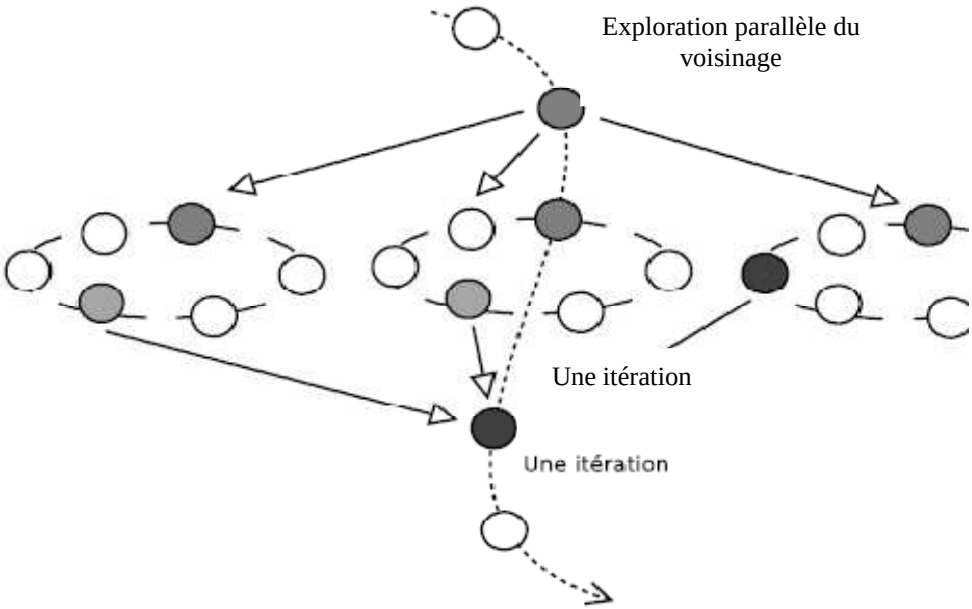


Figure 1.16. Le modèle parallèle d'évaluation du voisinage.

3.3.3. Une évaluation parallèle du mouvement

Ce modèle est efficace lorsque l'évaluation d'un seul mouvement est coûteuse. La fonction d'évaluation est divisée en un ensemble des nouvelles fonctions partielles. Il faut aussi définir un opérateur d'agrégation pour cette fonction globale.

4. Hybridation des métaheuristiques

Dernièrement et dans le domaine de l'optimisation combinatoire, les meilleurs résultats pour de nombreux problèmes de grande taille ont été obtenus par des métaheuristiques hybrides. L'utilisation des aspects complémentaires existants dans

différents types des métaheuristiques constituent des algorithmes de résolution hybrides très efficaces et robustes qui peut servir à une meilleure solution [09].

4.1.Taxinomie des métaheuristiques hybrides

Deux classifications existent dans la littérature pour les métaheuristiques hybrides : classification hiérarchique et à plat.

4.1.1. Une classification hiérarchique

Selon cette classification, on distingue l'hybridation dite de « haut » et de « bas » niveau. L'hybridation de bas niveau s'applique à la composition interne d'une métaheuristique, une fonction ou un mécanisme interne d'une métaheuristique est remplacé par une autre métaheuristique. Ce n'est pas le cas pour l'hybridation de haut niveau, la structure des métaheuristiques n'est pas modifiée et aucune relation n'existe entre leurs mécanismes internes [09].

Les deux types des métaheuristiques hybrides : de « haut » et de « bas », peuvent être en mode « relai » ou « Co-évolutionnaire ». En mode « relai », les métaheuristiques hybridées sont exécutées en séquence, les unes après les autres. Pour l'hybridation « Co-évolutionnaire », des modèles coopératifs d'optimisation sont impliqués, plusieurs processus s'exécution concurremment. Ainsi quatre classes hiérarchiques sont distinguées pour les métaheuristiques hybrides :

- La classe « LRH » (Low-level Relay Hybrid)
- La classe « LCH » (Low-level Co-evolutionary Hybrid)
- La classe « HRH » (High-level Relay Hybrid)
- La classe « HCH » (High-level Co-evolutionary Hybrid).

4.1.2. Une classification à plat

Selon cette classification et dans un premier point de vue, on distingue l'hybridation dite homogène de celle dite hétérogène : l'hybridation homogène est un couplage de métaheuristiques de même type, généralement avec des différents paramètres. Pour l'hybridation hétérogène, différentes types métaheuristiques sont utilisées.

D'un autre point de vue, nous distinguons l'hybridation globale et partielle : dans le premier cas les métaheuristiques impliqués dans des hybridations « globales » considèrent tous l'espace décisionnel. Dans le deuxième cas : l'hybridation partielle, le problème à résoudre est décomposé en sous-problèmes. Chacune des méthodes de résolution hybridées est associée à l'exploration d'un sous-espace dépendants les uns des autres, les métaheuristiques hybridées doivent coopérer afin de prendre en compte les contraintes

locales des chaque sous ensemble afin de construire une solution globale au problème traité.

Un troisième point de vue distingue l'hybridation avec une fonction générale de celle avec fonction spécialiste. L'hybridation générale désigne que tous les algorithmes impliqués visent à résoudre le problème d'optimisation, dans le cas d'une hybridation spécialiste les algorithmes mis en œuvre sont dédiées chacune à différents problèmes (Figure 1.16) [09].

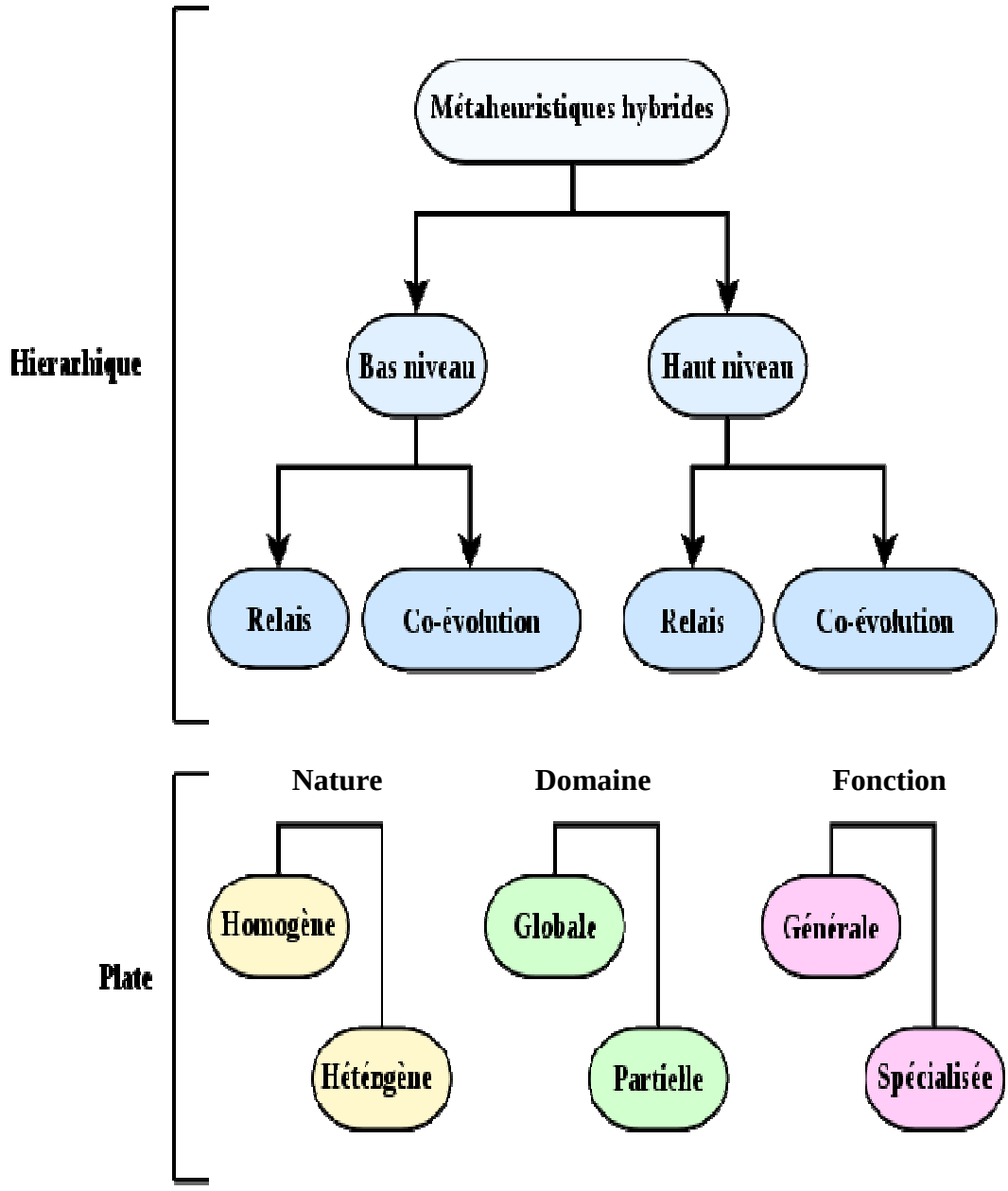


Figure 1.17. Taxomomie des modèle hybride pour les métaheuristiques.

5. Grille de calcul

Ces dernières années, le monde informatique a connu une grande évolution technologique des ressources informatiques, concernant la puissance de calcul et la capacité de stockage avec des prix abordables. Dans ces bonnes conditions, les systèmes distribués et parallèles sont aussi évolués grandement, commençant par les supercalculateurs et machine multiprocesseurs, passant aux différentes architectures : vectorielles, à mémoires partagées ou distribuées [21]. Dans un même temps, avec la disponibilité des masses d'ordinateurs, la baisse des prix et le développement des protocoles de communication inter-machines, l'idée de connecter des machines entre eux, est apparue [25], dans cet aspect le monde informatique a connu les réseaux de stations de travail (NOW) et les grappes de stations de travail (COW) [21]. Maintenant et depuis le début des années 2000 et avec la popularité d'internet, une nouvelle technologie appelée « grilles de calcul » trouvent une grande place dans le domaine de recherche en informatique.

5.1.Définition des grilles de calcul

Le terme « grille de calcul » est la traduction de grid computing. Ce nom est tiré de la nomination du réseau de distribution électrique appelé en Anglais electrical power grid. Où le principe est qu'un utilisateur consomme de l'électricité sans connaitre la façon ou de l'endroit où elle a été produite. Par analogie, l'idée du grid computing est de mettre des ressources à disposition des utilisateurs pour qu'ils profitent des puissances de calcul et de stockage sans savoir la source de cette puissance [26].

Beaucoup de définitions dans la littérature ont été données aux grilles de calcul, ces définitions sont fixées suivant l'objectif et le cadre dans lesquels on va entamer des travaux sur ces grilles, Dans notre cadre de travail sur l'optimisation combinatoire parallèle hybride, on définit une grille comme un ensemble de ressources de calcul reparties possédant les caractéristiques suivantes [21], [27].

Multi-domaine d'administration :

Plusieurs domaine d'administration existent, sur lesquels sont réparties les ressources, les utilisateurs et les fournisseurs de ces ressources sont identifiés préalablement, ainsi le problème de sécurité est significativement réduit, ça n'empêche pas que le problème d'intrusion de pare-feu réside encore et beaucoup de solutions ont été proposé pour sa résolution.

- ***La grille est hétérogène*** :

En ce qui concerne les ressources matériels et logiciels, elles peuvent être de différentes natures et appartenant à différentes organisations, l'utilisation des nouveaux standards tel que XML peut remédier à ce problème. Le calcul de la performance des applications implémentées constitue un autre problème à résoudre.

- ***La grille est extensible***

La grille est conçue de façon à permettre l'augmentation du nombre de machines disponibles ainsi que le nombre d'interconnexions de machines, les applications déployées sur les grilles doivent supporter le passage à l'échelle et l'extensibilité.

- ***La grille est dynamique***

La grille est une infrastructure avec pannes, en plus les ressources peuvent être à tout moment disponibles ou non, c'est-à-dire allouées aux utilisateurs ou libres selon les besoins d'utilisation, on est donc confronter à un problème de découverte dynamique de ressources, de tolérances au pannes, de synchronisation, etc.

5.2.Les couches d'une grille de calcul

Le développement des applications sur grille de calcul nécessite la prise en compte de ces caractéristiques, et aussi la régularisation des problèmes posés dans chacune d'elles [21]. Pour cela les intergiciels de grilles sont utilisés, la Figure 1.17 représente les couches d'une grille afin de comprendre le rôle de chacune et de saisir celui de l'intergiciel.

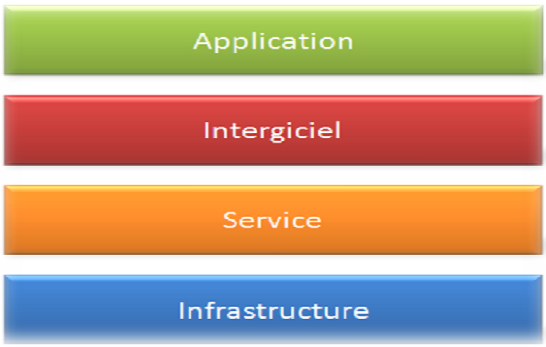


Figure 1.18. Modèle en couche des grilles.

5.2.1. La couche infrastructure

C'est la couche la plus basse du modèle, où les ressources vont être partagées via la grille. Ces ressources peuvent être physiques (des processeurs, des disques, des bases de données, le réseau) ou logiques : un système distribué de fichiers [26].

5.2.2. La couche service

Les services offerts par la grille de calcul peuvent être classifiés en quatre classes :

- La sécurité ;
- La gestion des données ;
- La gestion de l'exécution ;
- Et l'information : découverte des services et des ressources.

5.2.3. La couche intergiciel

C'est la couche intermédiaire entre les applications qui utilisent les services et l'infrastructure (ressources). Cette couche facilite aux applications l'accès aux différentes ressources de la grille en masquant les difficultés liées à leur gestion.

A ce niveau plusieurs problèmes liés à l'utilisation des grilles sont résolus, tels que la tolérance aux pannes, l'ordonnancement, la granularité, etc. On va citer quelques intergiciels et les problèmes qu'ils peuvent résoudre :

- ***Condor*** : dédié au calcul haut débit (High-Throughput Computing) permettant l'ordonnancement de nombreuses tâches indépendantes. Ce système met en œuvre des mécanismes pour la découverte de ressources, l'ordonnancement, l'accès aux données, la réservation des ressources, la sécurité et la gestion des droits d'accès.
- ***Globus*** : dédié au calcul haute performance (High- Performance Computing) permettant le développement incrémental d'outils et d'applications sur grilles [28] : support de nombreux langages, modèles de programmation, outils, applications, etc. Globus reste néanmoins très difficile d'accès. Son usage doit être motivé par l'implantation de grilles reliant les ordinateurs les plus puissants des plus grands centres de calcul à l'échelle internationale.

5.2.4. La couche application

Se sont les applications qui utiliseront les ressources et les services sur la grille de calcul. Ces applications peuvent être classifiées en plusieurs domaines :

- ***Super Calcul virtuel distribué à grande échelle*** : (Distributed virtuel supercomputing) ce type d'applications bénéfice d'un nombre beaucoup plus important de ressources de calcul qui lui réduit significativement le temps d'exécution des applications, permettant ainsi la résolution des problèmes non résolus auparavant [26] [21].
- ***Calcul haut débit*** : (High-Throughput Computing) : ces applications ont besoin d'augmenter la rentabilité au maximum des ressources en récupérant les cycles

processeurs non utilisés quand celui-ci est inactif [26], dans ce cas le temps d'exécution est réduit automatiquement.

- ***Traitement massif de données*** (Data maining) : pour ce type d'applications, le but est d'extraire de nouvelles informations à partir de grandes bases de données et librairies numériques distribuées géographiquement. Généralement, ces types de traitement sont gourmands en puissance de calcul et en débit de transfert de données.
- ***Calcul à la demande :*** ce type d'applications utilise la grille dans le but de satisfaire des besoins à court terme en ressources, tels que le temps de calcul, des logiciels, des données, etc. Ces besoins peuvent être non satisfaits en local pour des raisons de disponibilité, financière ou de rentabilité.
- ***Travail collaborative :*** le but des applications collaboratives est de permettre et favoriser les interactions entre les personnes. Elles sont souvent structurées sous forme d'espaces virtuels partagés entre les utilisateurs. La plupart de ces applications permettent de partager des ressources comme par exemple des données ou des simulations. Elles partagent alors bien souvent des caractéristiques avec les autres grands types d'applications décrites précédemment.
- ***Travail multimédia :*** ce genre d'application vise à utiliser un type particulier des services de grille qui est le support multimédia tel que la vidéo conférence.

5.3.Notre cadre d'utilisation des grilles de calcul

Dans le cadre des métaheuristiques parallèles hybrides pour l'optimisation combinatoire, on se situe sur deux volets d'application sur grille de calcul : le calcul distribué et le calcul haut débit. On vise à distribuer le calcul et profiter au maximum des ressources disponibles pour minimiser le temps d'exécution et augmenter la taille de l'espace de la recherche. Il s'agit d'une gridification des modèles parallèles des métaheuristiques à solution unique et à population de solutions. Il faut repenser à ces modèles en prenant en compte les caractéristiques des grilles de calculs.

Plusieurs implémentations, basées sur l'utilisation des intergiciels de grilles, ont été publiées, et ont montrés que la gridification des algorithmes d'optimisation combinatoire hybride est facilitée par les intergiciels des grilles, mais elles ont montrées aussi que la connaissance des concepts du parallélisme est indispensable. Pour cela les plateformes logicielles dédiées aux métaheuristiques parallèles hybrides sont utilisées. Sébastien CAHON dans sa thèse de doctorat soutenue le 1 juillet 2005, a comparé entre les

plateformes logicielles des métaheuristiques parallèles hybrides et il a dressé le tableau dans la Figure 1.18 :

Cette étude de comparaison est basée sur plusieurs critères ;

- Les métaheuristiques que supporte la plateforme logicielle ;
- Les possibilités d'hybridation des métaheuristiques ;
- Les possibilités des parallélismes ;
- Le langage de programmation ;
- Et les supports de communication entre les processus qui s'exécutent en parallèle.

	RL	AE	Hybrid.	Parall.	Langage	Support. Para.
ECJ	-	+	AE / AE	Coop. insulaire	Java	Threads Sockets TCP / IP
D-BEAGLE	-	+	AE / AE	F-T	C++	Sockets TCP / IP
J-DEAL	-	+	AE / AE	F-T	Java	Sockets TCP / IP
EasyLocal ++	+	-	RL / RL	-	C++	-
Localizer ++	+	-	RL / RL	-	C++	-
MAFRA	+	+	AE / AE RL / RL	-	Java	-
DREAM	-	+	AE / AE	Coop. insulaire	Java	Threads Sockets TCP / IP
MALLBA	+	+	AE / RL AE / AE RL / RL	Toutes	C++	Netstream MPI
EO	-	+	-	-	C++	-
ParadisEO	+	+	AE / RL AE / AE RL / RL	Toutes	C++	Threads MPI / PVM Condor / MW

Figure 1.19. Comparaison des plateformes de conception des métaheuristiques [9].

L'étude comparative de différentes plateformes conçues pour les métaheuristiques, nous a mené à s'intéresser à la plateforme Paradiseo, elle supporte les deux modèles des métaheuristiques : à solution unique et à population de solutions, en plus, notre recherche nous a permis de découvrir d'autres caractéristiques avantageuses pour les métaheuristiques parallèles et hybrides auxquels Paradiseo offre plusieurs mécanismes pour différentes architectures. On présentera cette plateforme dans la section suivante.

6. La plateforme Paradiseo

Paradiseo est une plateforme logicielle disponible en open source, elle est orientée Framework, portable sur différents systèmes d'exploitation : Windows, Unix et MacOs, elle est dédiée à la conception des métaheuristiques à population de solutions et aussi à solution unique. Paradiseo offre aussi les outils et les mécanismes pour la conception des métaheuristiques parallèles, distribuées et hybrides.

6.1.Les caractéristiques de la plateforme Paradiseo

La plateforme Paradiseo est caractérisée par la réutilisation maximale de code et de conception, offerte par la séparation conceptuelle claire et maximum entre méthodes de résolution et les problèmes traités. La partie générique des méthodes est fournie par la plate-forme et la partie spécifique au problème doit être fournie par le développeur. Cette séparation permet aux utilisateurs de passer plus de temps sur la compréhension de leurs problèmes que sur les méthodes de leur résolution.

Paradiseo est basée sur le principe d'Hollywood : ne nous appelez pas, nous vous appellerons. A l'inverse des librairies, c'est le code de la plate-forme qui appelle celui du programmeur. Puis, la flexibilité et l'adaptabilité en facilitant l'ajout et la modification de méthodes de résolution par simple spécialisation des composants ou classes fourni(e)s. Ensuite, l'utilité, en ce sens que Paradiseo fournit un large éventail de méthodes de résolution, de modèles parallèles et de mécanismes d'hybridation. Puis, la portabilité puisque Paradiseo permet un déploiement sur différents types de plates-formes dédiées ou non, à mémoire distribuée ou partagée, etc. Enfin, l'accès transparent à la performance et à la robustesse en intégrant tous les modèles parallèles et mécanismes d'hybridation présentés, exploitables de manière transparente.

6.2.Les modules de base de la plateforme Paradiseo

Paradiseo se compose essentiellement de quatre modules indépendants et complémentaires (Figure 1.19) respectivement désignés par :

- Paradiseo –EO« EO : Evolving Object »: offre les classes et les mécanismes pour la conception des métaheuristiques à population de solutions.
- Paradiseo-MO« MO : Moving Objects »: offre les classes et les mécanismes pour la conception des métaheuristiques à solution unique.
- Paradiseo-MOEO« MOEO : Multi-Objective Evolving Objects »: offre les classes et les mécanismes pour la conception des métaheuristiques pour l'optimisation multi-objectif.
- Paradiseo-PEO « PEO : Parallel Evolving Objects »: offre les classes et les mécanismes pour la conception des métaheuristiques parallèles, hybrides et aussi distribuées. Ce module n'a pas de version pour Windows.

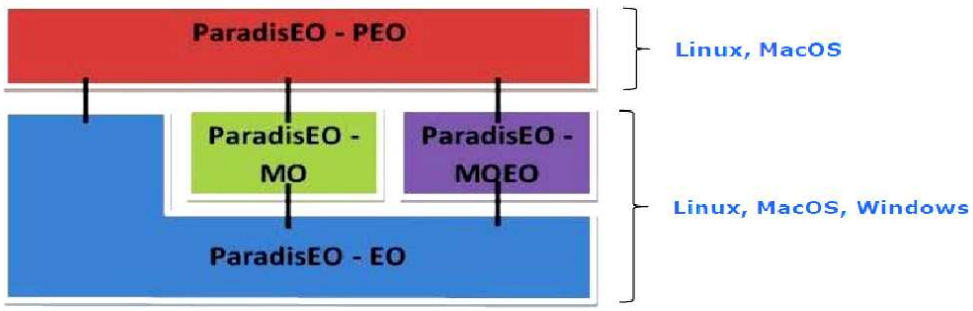


Figure 1.20.Les modules de Paradiseo.

6.3.Paradiseo et grille de calcul

La gridification des méthodes d'optimisation parallèles hybrides signifie la prise en compte des caractéristiques des grilles de calcul, c'est-à-dire la résolution des problèmes inhérents notamment de sécurité, de tolérance aux pannes, de minimisation des coûts de communication et de mesure des performances dans les contextes hétérogènes. Même si l'utilisation d'intergiciels permet de gérer une partie de ces problèmes de manière transparente, ces derniers ne répondent pas totalement aux besoins des applications, en plus leur utilisation exige la connaissance des concepts du parallélisme surtout sur grille de calcul souvent non métrisés par l'utilisateur, il est donc nécessaire de disposer d'un autre intergiciel entre l'application et l'intergiciel des grilles, le rôle de ce nouveau intergiciel est d'assurer le développement et le déploiement des composants d'optimisation combinatoire parallèle sur grille en masquant à l'utilisateur l'intergiciel permettant l'exploitation de celle-ci . Dans cet aspect Paradiseo a été couplée avec un intergiciel de grilles [30], et autres outils ont été utilisés pour garantir la communication entre les différent processus qui s'exécutent en parallèle : les bibliothèques MPI, PVM dans le cas de déploiement sur

réseaux de stations, les PThreads pour les machines multiprocesseurs à mémoire partagée SMP (Figure1.19).

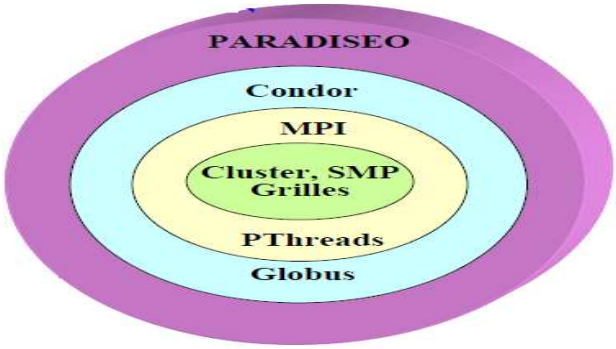


Figure 1.21. Paradiseo et grille de calcul.

7. Conclusion

Actuellement le grand défit est de résoudre un problème d'optimisation combinatoire générique de grande taille, où l'utilisation des métaheuristiques est recommandée. L'approche dite « A solution unique », en basant sur l'aspect de l'intensification, donne rapidement une bonne solution, mais risque être piégée par un optimum local. Ceci est évité avec l'aspect de la diversification fourni par l'approche connue « à population de solutions », cette dernière possède l'inconvénient que l'ensemble de solutions trouvées est juste une approximation de l'optimum. Ces deux notions : intensification/ diversification ne sont pas contradictoires mais complémentaires. Ce comportement complémentaire qui fait l'objet des nombreuses études [29], [30] a montré que l'hybridation des deux approches peut mieux guider le processus de la recherche et permet d'améliorer la qualité de solutions trouvées. En équilibrant entre l'exploration de l'espace de recherche et l'exploitation des régions prometteuses [09]. Les mécanismes d'hybridation s'avèrent très coûteux en temps d'exécution surtout pour les problèmes de grande taille. Cependant le recours au parallélisme est recommandé afin d'accélérer le processus de recherche. La technologie des grilles de calcul a permis de résoudre beaucoup de problèmes qui n'ont pas de solution auparavant.

Le déploiement des métaheuristiques parallèles hybrides sur grille de calcul nécessite la prise en compte des caractéristiques et les problèmes liés à l'utilisation de cette infrastructure, ceci est difficile à réaliser au niveau des applications, la maitrise des concepts du parallélisme sur les grilles de calcul tels que l'ordonnancement, la tolérance aux panne et la granularité, n'est pas facile pour tous les utilisateurs des grilles, pour ceci l'utilisation des intergiciels de grilles est indispensable.

CHAPITRE 2
LES PROBLEMES D'AFFECTATION QUADRATIQUE QAP ET Q3AP

1. Introduction

Ce chapitre est divisé en deux sections, la première concernera le problème d'affectation quadratique (QAP), un des problèmes d'optimisation combinatoire classique, qui possède de nombreuses applications pratiques, et beaucoup de méthodes de résolution ont été proposées dans la littérature pour sa résolution. Dans la deuxième section, notre intérêt sera un autre problème qui constitue une extension du QAP, il s'agit de l'affectation quadratique en trois dimensions Q3AP. Ce problème est relativement plus difficile, le plus grand problème résolu en séquentiel pour ce problème NP-difficile, est de taille n=13, en environ 7 jours par P. Hahn et al, et son parallélisme a été soumise à l'équipe Opale.

2. Le problème d'affectation quadratique

Le problème d'affectation quadratique (PAQ) ou en anglais quadratic assignment problem (QAP), est un problème classique en optimisation combinatoire, il a fait l'objet de nombreuses publications et est un cas d'étude académique [31]. Nous aborderons sa définition et ses applications.

Ce problème a été présenté pour la première fois par Koopmans et Beckmann en 1957, il est considéré comme un modèle mathématique pour le placement d'activités économiques indivisibles [32], [33]. Il consiste à déterminer le meilleur placement d'activités sur un ensemble de positions. Afin de déterminer la qualité d'une solution, plusieurs mesures sont utilisées : le coût des échanges qui est une fonction du flux (f) entre les activités, les distances (d) parcourues, et le coût d'installation (b) de l'activité sur une position donnée. Ces valeurs sont regroupées dans trois matrices : F la matrice des flux, où l'on trouve les valeurs des flux f entre les différentes activités, D la matrice des distances, où l'on trouve la distance d entre les différentes positions, B la matrice des coûts d'installation, où l'on trouve le coût b pour placer toutes les activités sur chacune des positions.

2.1.Présentation et formulation du QAP

Le problème proposé par Koopmans et Beckmann peut se présenter sous la forme d'un ensemble d'activités que l'on doit placer sur un ensemble de positions. Ces deux ensembles étant de même taille (n), les matrices F des flux, D de distances, et B des coûts d'installation sont symétriques, avec zéro sur la diagonale et aucune valeur négative pour les trois matrices. Une solution est une permutation de taille n, qui donne pour un élément (i) la position qui lui est attribuée (Φ (i)). L'objectif du problème est de placer toutes les activités avec un coût global minimal, ce dernier étant égal à la somme des coûts d'échange et d'installation pour toutes les activités dans les positions sélectionnées, il est donné par la formule mathématique numéro (1) [31], [33], [34], [35], [36], [37].

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij} d_{\Phi(i)\Phi(j)} + \sum_{i=1}^{n} b_{i\Phi(i)} \quad \ldots\ldots\ldots\ldots (1).$$

2.2.La version simplifiée du QAP

La version simplifiée est celle qui est adoptée pour les différents algorithmes que nous verrons, et qui est également la plus fréquemment utilisée. Il s'agit d'une simplification de la méthode proposée par Koopmans et Berkman. Ils ne tiennent pas compte du coût d'implantation, seuls sont considérés les coûts engendrés après l'implantation des activités sur les positions disponibles. La formule du coût global à minimiser sera donnée par la formule mathématique numéro (2).

$$\sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij} d_{\Phi(i)\Phi(j)} \quad \ldots\ldots\ldots\ldots (2)$$

2.3.Autres formulations du QAP

Une version plus générale du QAP a été introduite par Lawler [31], [34], [35], elle est donnée pour les problèmes dont lesquels ont peut mettre $C_{ijkl} = f_{ik} d_{jl}$, cette formulations mathématiques est équivalentes à la formulation donnée au dessus, elle permet différentes approches de résolution. Dans cette version, on a le coût de placement d'une activité i sur une position k, et une autre activité j sur une position l est de $C_{ijkl} = f_{ik} d_{jl} = C_{ij\Phi(i)\Phi(j)}$. La formule du coût global à minimiser sera donnée par la formule numéro (3):

$$\sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij\Phi(i)\Phi(j)} \quad \ldots\ldots\ldots\ldots (3)$$

2.4.Autre présentation du problème

Une solution du QAP peut être présentée par une permutation comme elle peut être présentée par une matrice carrée X dite matrice d'affectation, et la formule du coût global à minimiser sera donnée par la formule numéro (4) :

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} C_{ijkl} X_{ij} X_{kl} \quad \ldots\ldots\ldots (4)$$

$$X_{ij} = \begin{cases} 1 & \text{si l'activité i est affectée à la position j} \\ 0 & \text{si non} \end{cases}$$

Pour cette présentation, les deux contraintes : un site ne peut contenir qu'une seule usine, et une usine n'est affectée que sur un seul site, seront présentées par les expressions suivantes :

$$\begin{cases} \sum X_{ij} = 1 \text{ pour } j = 1 \ldots N \\ \sum X_{ij} = 1 \text{ pour } i = 1 \ldots N, \\ \text{Pour tous i, j, k, l : } i \neq j \text{ ou } k \neq l \text{ sinon } C_{iikk} := f_{ii} d_{kk}. \end{cases}$$

2.5.Applications du QAP

Ce problème trouve de nombreuses applications concrètes comme [31] :

- Le câblage de tableaux de bord : il s'agit de déterminer le schéma de câblage qui va minimiser les longueurs de fils utilisées.
- Le placement des caractères pour des machines à écrire : il s'agit de déterminer la place des caractères sur un clavier qui minimisera les distances parcourues, donc la durée pour la saisie d'un groupe de textes défini sera aussi minimiser.
- Le placement de services dans un hôpital : ici on cherche à minimiser la distance parcourue par les malades entre les différents services.
- L'ordonnancement de lignes de production parallèle.
- L'organisation dans une équipe de course de relais.
- L'analyse de réactions chimiques pour des composées organiques.

Pour illustrer ce problème, regardons deux exemples ; le premier donne la forme d'un problème concret, qui peut se modéliser par une instance du QAP, il s'agit d'implantation d'un ensemble des usines. Le second est un exemple pédagogique.

2.5.1. L'exemple de l'implantation d'usines

Un exemple du QAP pour le choix d'implantation d'usines est présenté dans la figure. 2.1. Le problème exposé consiste à implanter n usines sur n sites, une et une seule usine doit être placée sur chaque site. Les distances inter-sites sont connues. La distance entre deux sites quelconques i et j vaut dij. Les usines une fois implantées devront échanger des produits entre elles lors de leur fonctionnement. Le flux passant d'une usine k à une autre usine l est quantifié par la valeur f_{kl}. L'objectif dans ce cas est de déterminer l'implantation qui permettra par la suite de minimiser le coût total de transit entre les usines (ces coûts étant proportionnels à la quantité transportée et à la distance parcourue) [34], [31].

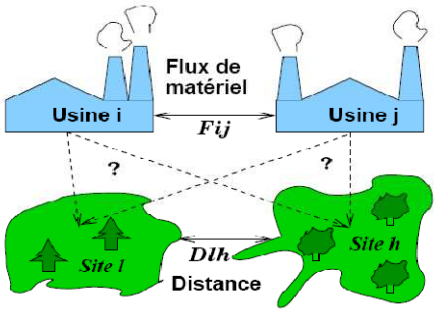


Figure 2.1. Illustration du QAP [22].

2.5.2. Un exemple pédagogique

Pour cet exemple présenté dans [31], on a quatre objets géométriques. Il est possible de les placer dans quatre emplacements distincts. Ces quatre figures communiquent de façon bidirectionnelle entre elles, chaque objet communique avec celui qui le précède et celui qui le suit. Ce problème peut être formalisé sous la forme des deux matrices : la première est la matrice des flux F ; et la seconde est la matrice des distances D. Dans notre exemple, les emplacements constituent un carré (Figure 2.2), les distances des diagonales sont les plus longues. Un exemple d'une solution aléatoire est présentée, le coût des échangés est ici égal à 5. Une des solutions optimales, (qui sont au nombre de huit), est également indiquée, son coût étant égal à 3. Dans cet exemple la position d'un objet est donnée par l'indice du tableau où il est placé ; par exemple le premier objet est à la position 1.

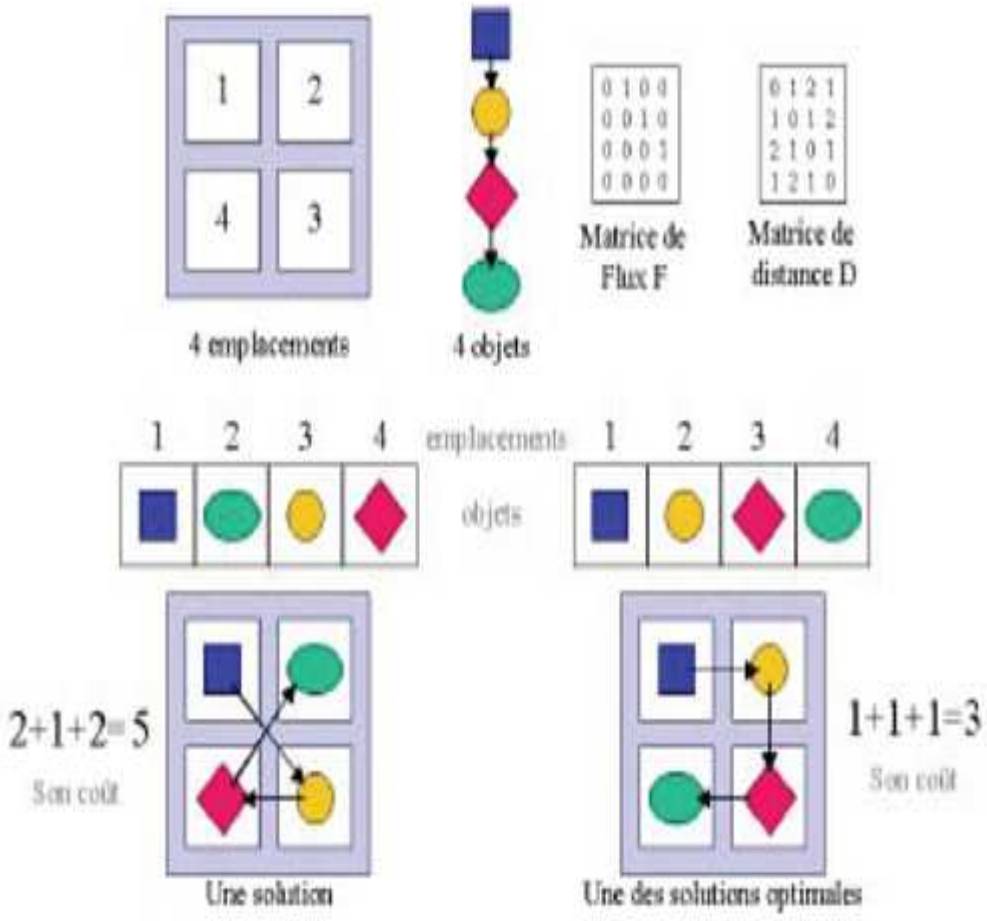


Figure 2.2. Illustration de l'exemple pédagogique.

2.6.Instances du QAP

La QAPLIB est un regroupement d'instances du QAP, qui ont été réunies par Burkard, Karisch et Rendl [29]. Cette librairie permet d'avoir un ensemble de problèmes sur lesquels il est possible de comparer les heuristiques. Les instances qui composent la QAPLIB sont désignées par un nom qui 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. Ainsi, bur26a a été créée par R.E. Burkard et J. Offermann, et correspond à un problème contenant 26 sites et 26 objets. Dans cet exemple, le nom est complété par une lettre qui sert à différencier les instances de taille identique pour un même auteur. Des informations sur les instances sont disponibles dans la librairie. Il est possible d'y trouver :

- la solution optimale quand elle est connue ;
- la meilleure solution trouvée accompagnée de la méthode qui l'a obtenue ;
- et une valeur planchée pour le coût ;

Les instances sont classifiées en trois catégories distinctes qui se différencient par la nature de leurs matrices de flux et de leurs matrices de distances :

- les instances uniformes sont générées aléatoirement, elles sont donc très peu structurées ;
- les instances réelles ou assimilées sont très fortement structurées ;
- et les instances générées dynamiquement sur grille.

Voici un exemple d'instance de la librairie QAPLIB :

Instance	Taille	Distances	Flux
Tai25a	25	Uniforme	Uniforme
Tai100a	100	Uniforme	Uniforme
Nug30	30	Grille	Aléatoire
SKo64	64	Grille	Aléatoire

Tableau 2.1. Exemple d'instance QAP de la librairie QALIB.

2.7.Méthodes de résolution pour le QAP

Beaucoup de recherches ont été menées sur ce problème complexe à résoudre. La résolution complète d'une instance de QAP demande l'évaluation de toute les possibilités d'affectation qui sont au nombre de 20 ! = 2,36. 10^{18}, pour un problème de taille 20, ce qui représente, si on évalue un milliard de solutions à la seconde sur une machine, un temps de calcul de 28 158 jours. Ils existent quelques algorithmes de résolution exacte pour le QAP, parmi lesquels on peut citer la méthode Branch and Bound. Un problème de taille supérieure à 20 ne peut pas être résolu par ces méthodes avec un temps de calcul raisonnable, le temps de calcul étant un facteur refusable. Pour pallier ce défaut, de nombreuses heuristiques ont été développées, offrant de bonnes solutions pour un temps de calcul raisonnable.

Les méthodes qui ont été utilisées pour la résolution sont :

- les méthodes constructives ;
- les méthodes d'énumération limitées ;
- les méthodes d'optimisation locale ;
- la recherche Tabou ;
- le recuit simulé ;
- les algorithmes génétiques ;
- la méthode GRASP (greedy randomized search procedure
- la recherche par dispersion ;
- et les méthodes à base de colonies de fourmis.

3. L’affectation quadratique en trois dimensions

En anglais the Quadratic 3-dimensional Assignment Problem Q3AP, est la modélisation d'un problème stratégique qui vient d'une nouvelle application pratique en transmission de données sans fil où un schéma hybride ARQ (Automatic Repeat reQuest) est implanter 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. Ce problème NP-difficile est considéré comme une extension du QAP [34], [35], [36].

3.1.Exemples Q3AP d'implantation des usines

Cette problématique consiste à chercher la meilleure affectation possible des *n* usines et *n* managers sur les *n* sites parmi (n ! x n !) solutions possibles. Le problème est classé NP-difficile et devenu plus complexe avec les instances de grandes tailles.

Pour ce problème, n usines et n managers doivent être placés sur n sites. Deux sites l et h sont à une distance de d_{lh}. Deux usines i et j échangent un volume f_{ij} de matières. Deux managers m et k ont un taux d'échange de e_{mk} comme il est illustré dans la Figure2.3. Le coût de placer une usine i et un manager m sur le site l et une autre usine j avec un autre manager k sur le site h, est de $C_{ijlhmk} = e_{mk}.d_{lh}.f_{ij}$.

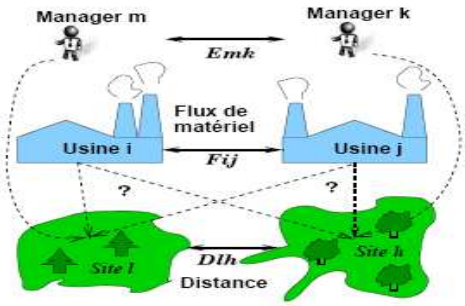


Figure2.3. Illustration du Q3AP [37].

3.2.Formulation du Q3AP

Afin de déterminer la qualité d'une solution, plusieurs mesures sont utilisées : le flux (f) des marchandises échangées entre les usines, les distances (d) parcourues entre les sites, le taux (e) d'échange d'information entre managers, et le coût d'installation des usines sur les sites donnés, ces mesure sont regroupées dans quatre matrices : F la matrice des flux, D la matrice des distances, E la matrice des taux d'échange d'information, et B la matrices des coûts d'installation. La résolution du problème Q3AP consiste à placer les usines et les mangers sur les sites disponibles avec un coût global minimal, ce dernier étant égal à la somme des coûts d'échange et d'installation des usines et des managers sur les sites sélectionnés, il est donné par la formule numéro (5) [35], [37], [38]:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N}\sum_{k=1}^{N}\sum_{n=1}^{N}\sum_{q=1}^{N} C_{ijpknq} X_{ijp} X_{knq} + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N} b_{ijp} X_{ijp} \quad \ldots\ldots (5)$$

Pour la version simplifiée du Q3AP, les coûts d'installation ne sont pas prisent en compte, et la formule de la fonction coût à minimiser sera donnée par la formule numèro (6) :

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{N}\sum_{k=1}^{N}\sum_{n=1}^{N}\sum_{q=1}^{N} C_{ijpknq}X_{ijp}\,X_{knq} \dots\dots (6)$$

$$\text{Où} \begin{cases} x \geq 0 \ \sum\sum X_{ijp}=1 \text{ pour } i=1\ldots.N \\ x \geq 0 \ \sum\sum X_{ijp}=1 \text{ pour } j=1\ldots.N \\ x \geq 0 \ \sum\sum X_{ijp}=1 \text{ pour } p=1\ldots.N \end{cases}$$

Une entre condition à prendre en compte, est que pour tous les termes de la fonction objective du Q3AP, objective ne doit pas contenir les termes $X_{ijp}X_{knq}$ où $i=k$ ou $j=n$ ou $p=q$. Si $i=k$ et $j=n$ et $p=q$ alors $x_{ijp}\,x_{knq}=x_{ijp}$, sinon $x_{ijp}\,x_{knq}=0$.

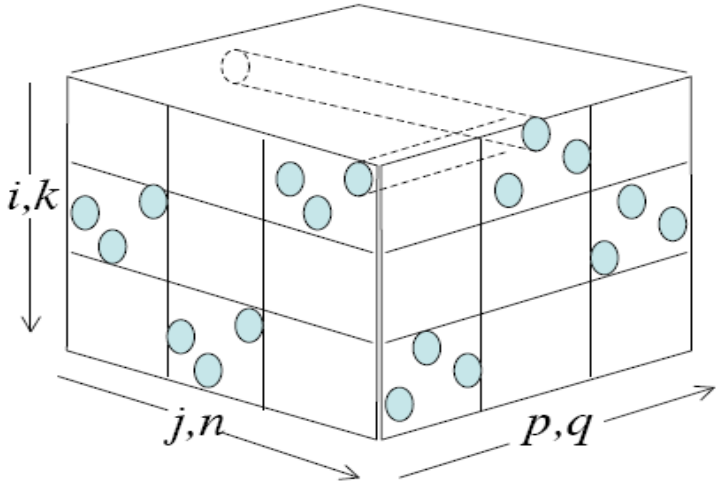


Figure 2.4. Présentation matricielle du Q3AP.

3.3.Présentation par permutation

Par similarité au problème QAP, une solution du problème Q3AP peut être présentée par deux permutations, Φ[i]=j signifie que l'usine j est affectée au site i, φ[i]=j et signifie que le manager j est affecté au site i, la formulation du problème sera donnée par la formule numéro (7) :

$$\sum_{i=1}^{N}\sum_{j=1}^{N} C_{i\Phi(i)\Phi(j)j\varphi(i)\varphi(j)} \dots\dots\dots\dots (7)$$

3.4. Le problème de transmission fiable des données sans fils :

Afin d'assurer la fiabilité des transmissions, deux grandes classes de mécanismes sont utilisées : les mécanismes réactifs ARQ (Automatic Repeat request) et les mécanismes proactifs FEC (Forward Error Correction). Dans les mécanismes réactifs, l'émetteur réagit à la signalisation d'une perte de paquet en retransmettant ce paquet. Cette signalisation peut être effectuée par l'émission d'acquittements positifs ACK (ACKnowledgement) ou négatifs NAK (Negative-AcKnowledgement). En ce qui concerne le fonctionnement des mécanismes proactifs, l'émetteur rajoute des paquets de redondance permettant au récepteur de récupérer des paquets perdus. Ces paquets de redondance sont calculés en utilisant des codes correcteurs d'erreurs [22]. Ces deux mécanismes sont souvent combinés en utilisant des acquittements pour ajuster la quantité de redondances des codes FEC. Une telle combinaison de mécanismes est appelée « ARQ hybride » ou HARQ (Hybrid Automatic Repeat reQuest). Le HARQ peut être caractérisé par certains paramètres tels que la synchronisation, l'adaptabilité, ainsi que la manière dont est faite la combinaison. Quand la relation temporelle entre la transmission originale et la (ou les) retransmission(s) est fixe, l'opération HARQ est dite alors « synchrone ». Si, par contre, les retransmissions sont programmées à n'importe quel moment après avoir reçu un ACK, on parlera alors d'opération « HARQ asynchrone ». On dit qu'un système HARQ est adaptatif si on peut réaliser des retransmissions en utilisant un autre type de modulation autre que celui qui a été utilisé pour la transmission originale. Par exemple, si la modulation QPSK est utilisée durant la première transmission d'un paquet et que celle-ci échoue, la 16-QAM peut être utilisée pour la retransmission de ce même paquet si les conditions du canal ou des ressources radio (puissance du signal et/ou codes de canalisation) changent entre la première transmission et les retransmissions qui se suivent. La résolution du Q3AP peut réduire Le **taux d'erreur** ou **B.E.R.**, abréviation de l'expression anglaise **Bit Error Rate**, relative au niveau d'atténuation et/ou de perturbation d'un signal transmis. Les séquences de bits sont affectées à des symboles QAM suivant le type de modulation utilisé (voir ANNEX B). A la réception les symboles sont décodées pour avoir les séquences de bits, on cherche la meilleure affectation des symboles aux séquences de bits qui minimise la fonction BER durant les deux transmissions [39].

Pour le protocole Hybride ARQ asynchrone, on a la première transmission, avec une première affectation des symboles aux séquences de bits, si une erreur est détectée, une retransmission est produite avec une deuxième affectation d'un autre code QAM aux mêmes séquences de bits. On cherche à déterminer l'affectation optimale des symboles QAM aux séquences de bit qui minimise le BER des deux transmissions, il s'agit d'une résolution du Q3AP.

- Première transmission d'un code binaire

  Usine ⟶ Symbole **QAM.**

  Site ⟶ Code binaire à envoyer.

  QAP : Trouver une affectation symbole –code binaire qui minimise le BER

- Retransmission du même message.

  Manager ⟶ Symbole **QAM.**

  Site ⟶ Code binaire à envoyer.

Q3AP : Trouver une nouvelle affectation qui minimise le BER pendant les deux transmissions [37].

3.5.Métaheuristique pour résoudre le Q3AP

Les algorithmes de recherche locale stochastiques sont les plus adaptés pour la plus part des problèmes d'optimisation combinatoire (Aarts and Lenstra, 1997; Hoos and Stutzle, 2004). Ces méthodes sont aussi les meilleures solutions pour les problèmes industriels avec fortes contraintes. Pour cela, ces algorithmes ont été adaptés pour résoudre le Q3AP: recuit simulé de Connolly (1990), la recherche tabou et les colonies de fourmis de Taillard (1991, 1998), et la recherche locale itérative de Stutzle (2006). Ces expériences on montré que ces algorithmes peuvent trouver des solutions optimales pour nombreuses instances QAP, et plus rapidement que les méthodes exactes. Par exemple la recherche locale itérative trouve la meilleure solution pour l'instance de Nugent de taille 30 dans 1.3 secondes CPU. Ces résultats sur QAP ont poussé les propositions d'adaptation de ces méthodes pour le Q3AP de taille large supérieure à 64. Bum-Jin Kim a adapté ces algorithmes, les instances du Q3AP sont générées a partir des instances du QAP en remplaçant la formule de calcul du coût $C_{ijkn}=F_{ik}*D_{jn}$, par la formule: $C_{ijpknq}=F_{ik}*D_{jn}*F_{ik}*D_{pq}$. Bum-Jin Kim a montré que la recherche locale itérative est la plus performante que les autres métaheuristiques, ces derniers ne peuvent pas garantir l'optimalité de la solution par rapport aux algorithmes exacts, mais elles peuvent trouver de

bonnes solutions proches de l'optimum dans un temps réduit, et même pour des problèmes de taille supérieure à 20, non résolus par les méthodes exactes [38].

4. Conclusion

À travers ce chapitre, on a présenté les deux problèmes QAP, et Q3AP, leurs domaines d'application et leurs méthodes de résolution. La résolution des deux problèmes n'est pas facile, l'explosion combinatoire était le problème majeur rencontré lors des différents essais de résolution existant dans la littérature. Maintenant, nos deux premiers chapitres, nous amènent à proposer un modèle parallèle hybride pour résoudre ce type de problèmes. L'aspect de l'intensification offert par les métaheuristiques à solution unique, et l'aspect de diversification offert par les métaheuristiques à population de solution, nous donne un espoir à d'atteindre une nouvelle valeur minimale pour la fonction d'évaluation, donc notre intérêt dans le chapitre suivant sera ce nouveau modèle.

CHAPITRE 3

CONCEPTION D'UN MODELE PARALLELE HYBRIDE A L'AIDE DE LA PLATEFORME PARADISEO

1. Introduction

Dernièrement, et dans le domaine de l'optimisation combinatoire, les meilleurs résultats pour de nombreux problèmes de grande taille ont été obtenus par des métaheuristiques hybrides [7]. L'utilisation des aspects complémentaires existants dans différents types des métaheuristiques peut servir à une meilleure solution. L'approche dite « A solution unique », en se basant sur l'aspect de l'intensification, donne rapidement de bonnes solutions, mais risque d'être piégée par un optimum local. Ceci est évité avec l'aspect de la diversification fourni par l'approche « à population de solution », cette dernière possède le désavantage que l'ensemble de solutions trouvées est juste une approximation de l'optimum .Cependant on voit que l'hybridation des deux approches peut mieux guider le processus de la recherche vers la solution optimale, le mécanisme du parallélisme constitue une puissance de calcul et peut accélérer le processus de l'optimisation. Afin de saisir les gains de cette hybridation parallèle, on propose une application de cette dernière sur l'affectation quadratique QAP, et l'affectation quadratique en trois dimensions Q3AP.

2. Notre proposition de solution

Pour notre problématique, on propose un modèle hybride où on va remplacer l'opérateur de l'algorithme génétique par l'algorithme Hill Climbing connu par sa bonne exploitation de l'espace de recherche , dans le but d'améliorer la qualité de la solution mutée dans la population actuelle, et rapprocher de plus en plus vers la solution optimale. D'autre part, on vise à résoudre un problème de grande taille dont le nombre de solutions possibles est élevé, notre intérêt sera donc de visiter le plus grand nombre possible de ces solutions afin d'augmenter la chance d'atteindre la solution optimale ou même une solution plus proche de l'optimale. Un autre facteur à prendre en compte, est que la phase de transformation de l'algorithme génétique (mutation, recombinaison, évaluation) est la plus couteuse en temps CPU. Dans cette situation, on voit que deux possibilités peuvent accélérer le processus de l'optimisation et améliorer la qualité de la solution trouvée :

1- Un algorithme hybride parallèle selon le modèle insulaire pour distribuer le processus de la recherche sur plusieurs sous populations indépendantes, sur chaque sous population un AG s'exécute et l'opérateur de mutation de cet algorithme est remplacé par le Hill Climbing, les sous populations communiquent entre elles à travers le mécanisme d'immigration appliqué avec un taux précis et avec une taille de l'ensemble des individus immigrés bien déterminée.
2- Un algorithme hybride parallèle de la phase de transformation de l'AG selon le modèle Maitre/esclave, et l'opérateur de mutation est remplacé par le Hill Climbing. Le modèle maitre esclave possède plusieurs avantages cités dans le chapitre 1, pour cela il a été choisi.

3. L'algorithme génétique classique pour le QAP

Le codage des individus de la population initiale constitue une étape préalable de tout algorithme génétique. Les étapes de selection, recombinaison, mutation, remplacement, et l'évaluation de la fonction objective sont les étapes de tout algorithme génétique.

3.1. Codage des individus (solutions)

Pour le problème QAP, chaque individus est représenté par une permutation, dans notre problématique le Q3AP, un individu est représenté par deux permutations Φ et φ de nombre de 1 à n, la première permutation correspond à l'affectation des usines sur les sites, et la deuxième correspond à l'affectation des managers sur les sites, c'est-à-dire que $\Phi[i] =$

j signifie que le manager j est affecté au site i, et φ[k]=p signifie que l'usine p est affecté au site k. Chacune des deux permutations sera représentée par un vecteur de taille n.

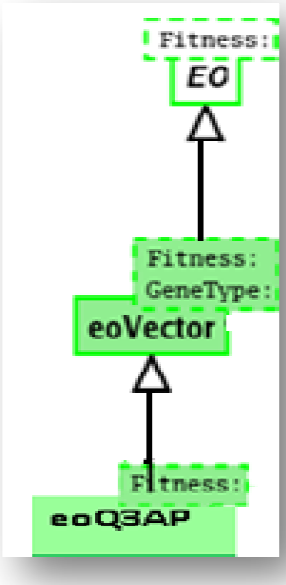


Figure 3.1. Présentation UML des classes de présentation des solutions

3.2.La fonction d'évaluation

La fonction d'évaluation d'une solution du Q3AP est présentée par la classe « eoEvalFunc ». La population des solutions est présentée par la classe « eoPopEvalFunc » comme il est illustré dans la figure 3.2.

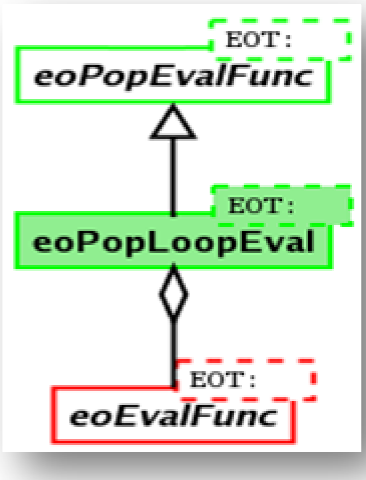


Figure 3.2. Présentation UML des classes d'évaluation d'une population.

3.3.Opérateur de sélection

Les individus de la population initiale sont soumis à une sélection par tournois. Motivée dans le chapitre1. Cette méthode est une agrégation des autres classes illustrées dans la figure 3.3.

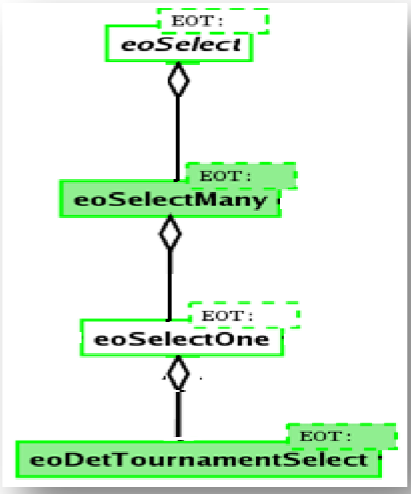


Figure 3.3. Présentation UML des classes de l'opérateur de sélection

3.4.Opérateur de recombinaison

Les individus sélectionnés seront recombinés deux à deux pour donner naissance à des nouveaux individus. Concernant le QAP, la recombinaison choisie est très simple. Pour produire un enfant, on copie les attributs communs aux deux parents. Les autres entrées sont choisies au hasard, ce processus s'effectue comme il est illustré dans la figure 3.4. Dans notre problématique le Q3AP, un individu enfant est présenté par une permutation de son premier père, avec une autre permutation de son deuxième père, ce processus est illustré dans la figure 3.5.

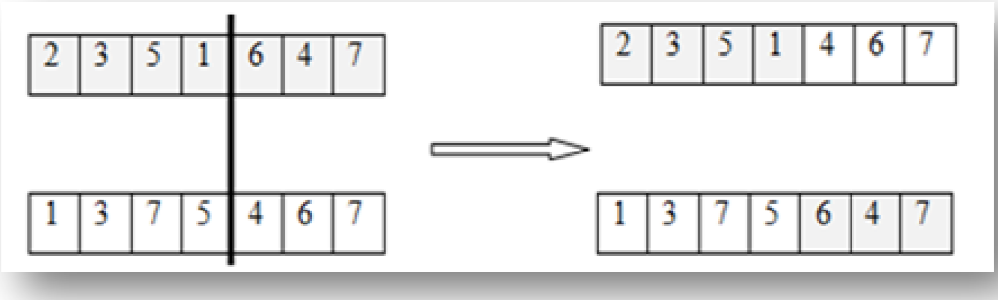


Figure 3.4. Présentation du processus de recombinaison pour le QAP.

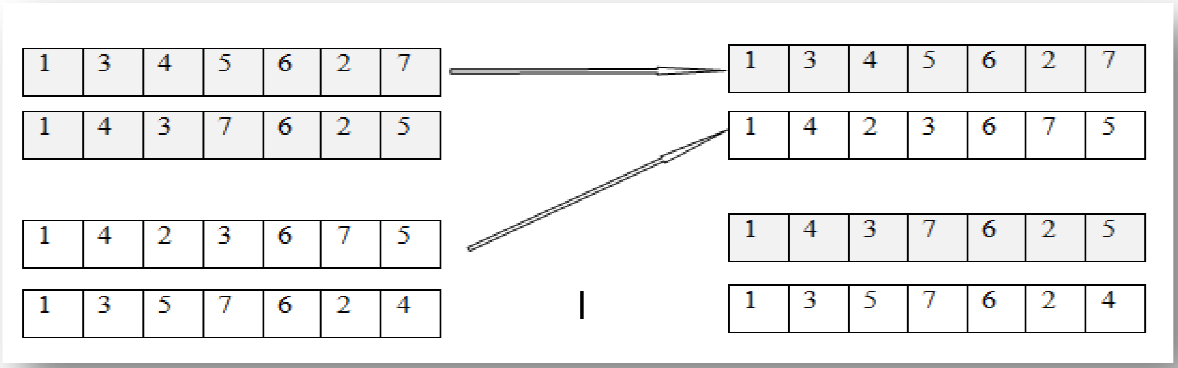


Figure 3.5. Présentation du processus de recombinaison pour le Q3AP.

3.5.Opérateur de mutation

Pour le QAP, et avec une petite probabilité, deux gènes dans un chromosome se permutent, cela signifie qu'on a changé l'affectation de deux usines. Dans le cas du Q3AP, une mutation est une transposition de deux éléments dans les deux permutations, si les deux gènes **Φ (i)** et **Φ(j)** sont transposées**,** les gènes **φ (i)** et **φ (j)** seront aussi transposées**.**

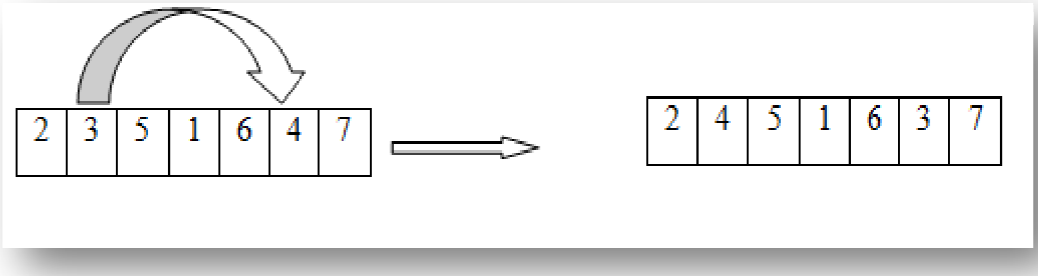


Figure 3.6. Présentation du processus de mutation pour le QAP.

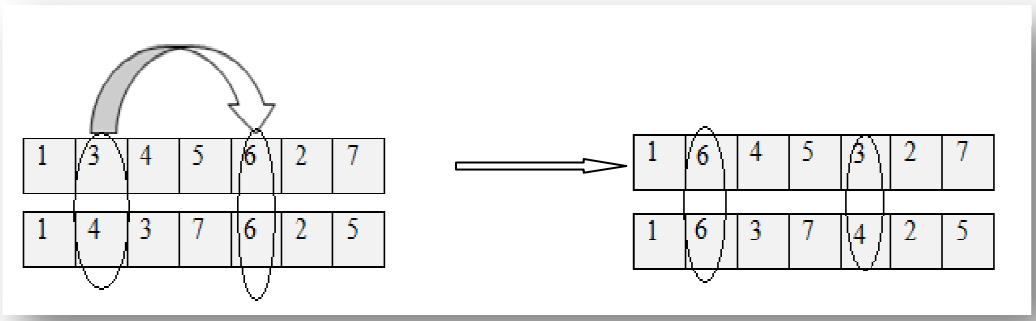


Figure 3.7. Présentation du processus de mutation pour le QAP

Les deux opérateurs, de recombinaison et de mutation est représenté par les deux classes eoQuadOp et EoMonOp respectivement, ces dernière constitue deux agrégations de la classe eoTransform (Figure3.7).

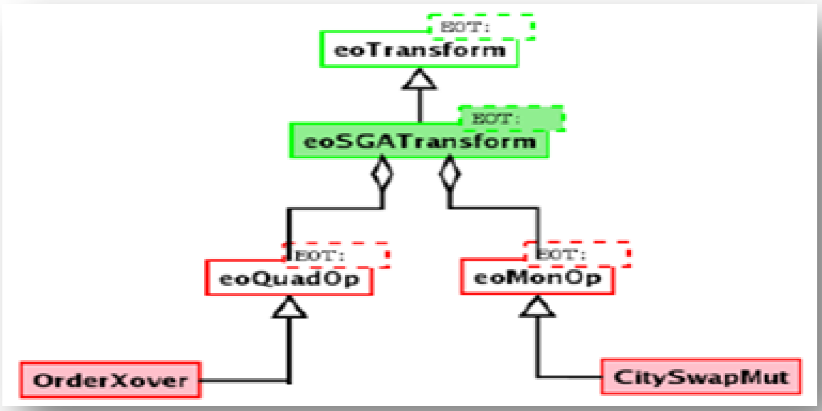


Figure 3.8. Présentation UML des opérateurs de recombinaison et mutation.

3.6.Opérateur de remplacement

En fonction de leurs fonctions d'évaluation, les individus les moins adaptés seront remplacés par les nouveaux individus, il s'agit d'un remplacement élitiste. La même présentation UML de cet opérateur pour les deux problématiques QAP et Q3AP (Figure3.7).

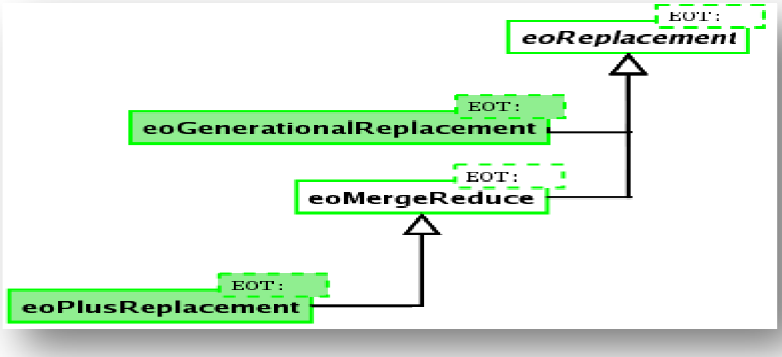


Figure 3.9. Présentation UML de l'opérateur de remplacement.

3.7.Le critère d'arrêt :

Le critère d'arrêt de l'algorithme génétique parallèle sera un nombre fixe d'itérations, ce critère est représenté par la classe « eoGenContinue » qui est une

agrégation de la classe « eoContinue », la figure 3.9 est une présentation UML du critère d'arrêt identique pour les deux problématiques QAP et Q3AP.

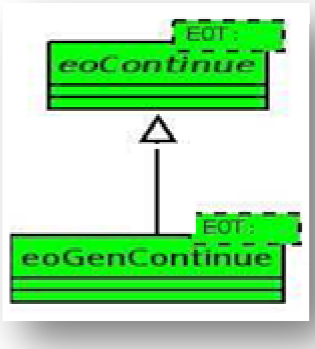


Figure 3.10 Présentation UML du critère d'arrêt.

4. Conception UML de l'algorithme génétique

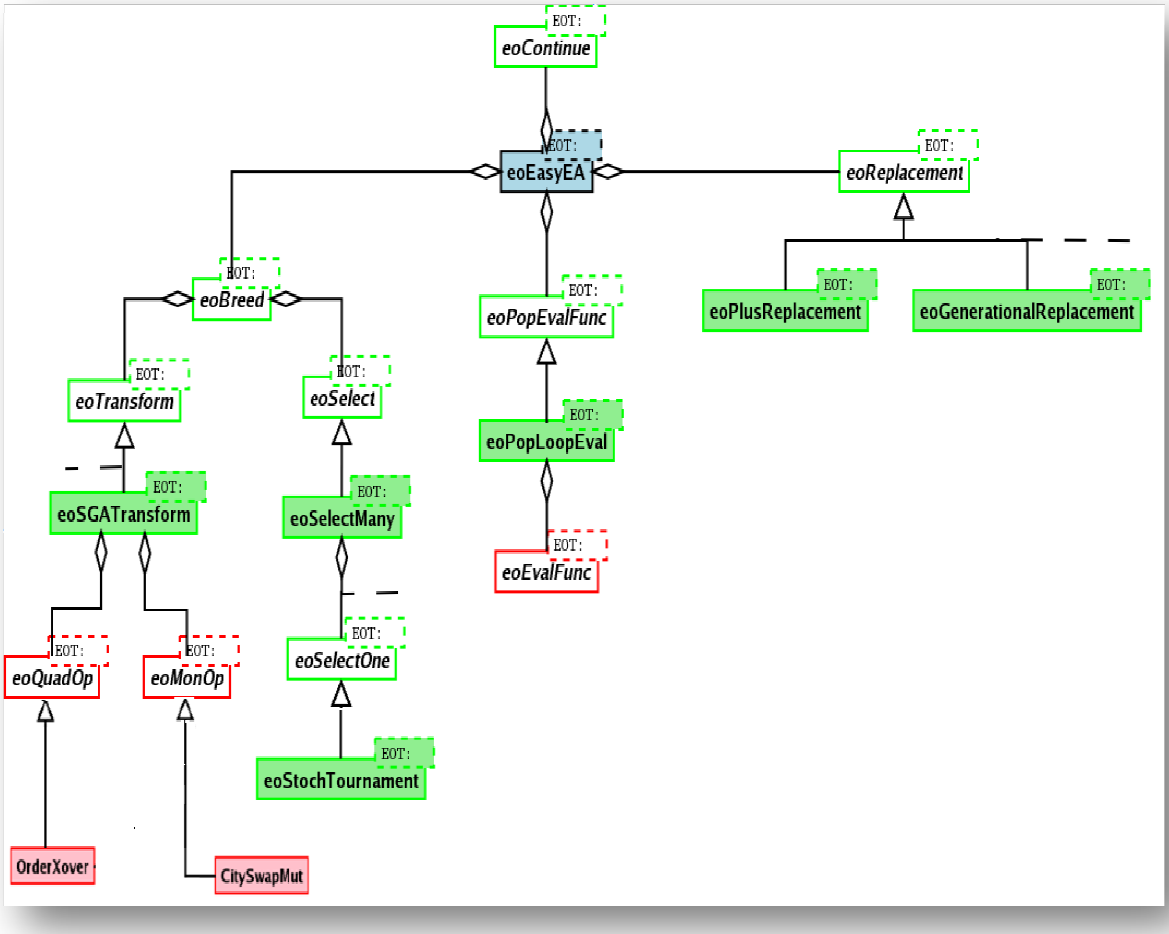


Figure 3.11. Présentation UML de l'algorithme génétique.

A partir des principales étapes constituant le schéma d'exécution itératif du modèle proposé, on a identifié les principaux composants associés aux opérateurs génétiques mis en œuvre. On distinguera les mécanismes dont le fonctionnement est indépendant du problème traité (la sélection « eoSelect », le remplacement « eoReplacement», le critère de décision de continuation « eoContinue ») des autres plus spécifiques (La génération d'une solution initiale aléatoire « eoInit », la fonction d'´evaluation « eoEvalFunc », les opérateurs de variation « eoQuadOp » et « eoMonOp »), etc.

5. L'algorithme Hill Climbing

Le principe de l'algorithme Hill Climbing consiste à choisir une solution initiale $\mathbf{s_0}$ possédant une fonction objective $\boldsymbol{F(s_0)}$ est choisie aléatoirement. Par la suite l'ensemble $\boldsymbol{V(s_0)}$ de solutions voisines de $\boldsymbol{s_0}$ est généré. Parmi l'ensemble du voisinage, une solution $\boldsymbol{s}$ appartenant à $\boldsymbol{V(s_0)}$, qui minimise tout les $\boldsymbol{F(s)}$, étant **s** appartient à $\boldsymbol{V(s_0)}$, sera sélectionnée. Si $\boldsymbol{F(s)}$ est inférieur à $\boldsymbol{F(s_0)}$, il faut réitérer ces étapes avec $\boldsymbol{s}$. Sinon on va prendre $\boldsymbol{s_0}$ comme solution optimale.

D'abord nous devrons définir notre mouvement. Dans le QAP, un mouvement est une transposition de deux éléments de la permutation. Pour le Q3AP il sera la transposition de deux éléments dans les deux permutations, si on a transposé $\mathbf{\Phi(i)}$ et $\mathbf{\Phi(j)}$, on doit transposer $\mathbf{\varphi(i)}$ et $\mathbf{\varphi(j)}$. Pour l'exemple d'implantation des usines, si deux usines sont permutées entre deux sites, les managers de ces deux sites seront aussi permutés.

A l'aide de la plateforme Paradiseo, l'algorithme Hill Climbing est représenté par la classe moHC (figure 3.8) qui englobe les cinq classes d'opérateurs suivantes :

- « moMveInit » : pour définir une classe d'initialisation des mouvements. En d'autres termes, pour indiquer quelles sont les deux premiers éléments à échanger pour réaliser un mouvement initial.
- « moMoveNext » : pour explorer tous les voisins d'une solution, nous avons besoin de savoir comment générer ce voisinage. Pour le QAP, le voisinage d'une permutation est l'ensemble de toutes les permutations que nous pouvons obtenir en échangeant tous les éléments. Le même principe pour le Q3AP, un voisinage est l'ensemble de tous les états engendrés par les mouvements possibles.
- « eoEvalfunc » : la fonction dévaluation compète d'une solution.
- « moMoveIncrEval » : il est possible de calculer la fonction évaluation d'une nouvelle solution à partir de la fonction évaluation de l'ancienne solution, et le

mouvement effectué sans faire une évaluation complète de la nouvelle solution. L'idée est de connaître la solution en cours et le passage que nous avons appliqué pour déduire la nouvelle valeur.

- « moMoveselect » : la stratégie de sélection qui est dans notre cas moBestImprselet c'est-à-dire choisir le meilleur voisin possédant la fonction coût minimale.

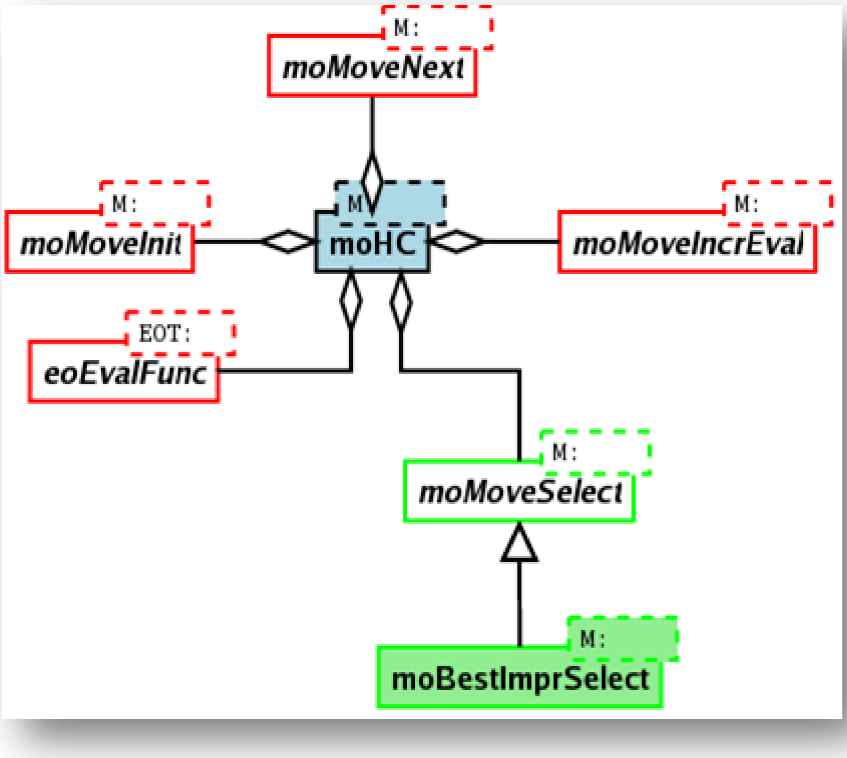


Figure 3.12. Présentation UML de l'algorithme Hill Climbing.

6. Le modèle parallèle insulaire

Selon un modèle insulaire, la population sera divisée en sous populations évoluant en parallèle et connectées sous une topologie donnée. Sur chacune des sous populations, l'AG s'exécute normalement. Les sous populations coopèrent entre elles à travers le mécanisme d'immigration vérifié au terme de chaque itération après le processus de remplacement comme il est illustré dans l'organigramme figure.

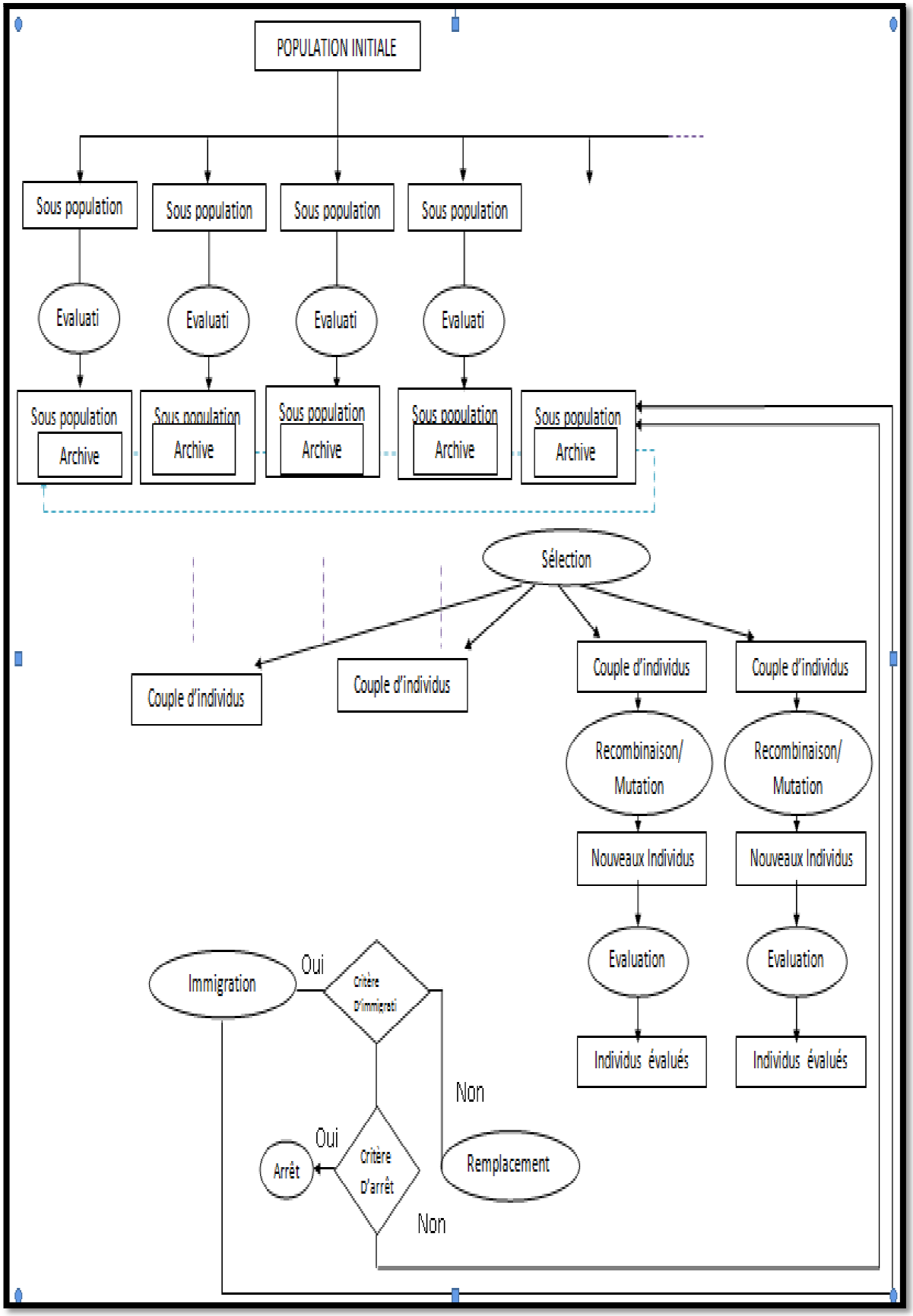


Figure 3.13. Organigramme du modèle parallèle insulaire.

Avant d'exécuter le mécanisme d'immigration une opération de sauvegarde de la population immigrée est effectuée grâce au mécanisme « eoCheckPoint », ce qui rend le modèle insulaire tolérant au panne. Par la suite plusieurs critères seront identifiés (Figure 3.12) :

- la topologie de connexion inter les sous populations du modèle insulaire : on va tester la topologie anneau motivée par sa simplicité d'implémentation, elle constitue un graphe cyclique avec un nombre limité de connexions (dans le cas contraire, les sous population se comportent comme une seule population globale). Cette topologie est présentée par la classe « eoRingTopology ».
- Le critère d'immigration à vérifier : dans notre cas l'immigration est synchronisée par un nombre fixe d'itérations de l'algorithme génétique : les meilleurs éléments seront choisis. Ce critère est présenté par la classe « eoContinue ».
- La stratégie de sélection des solutions qui seront immigrées, les meilleurs individus seront immigrés.
- La stratégie de remplacement des solutions qui seront écrasées par celles immigrées, les éléments les moins adaptés seront immigrés. Chaque nœud émit sont archive à son successeur et reçoit l'archive de son prédécesseur.

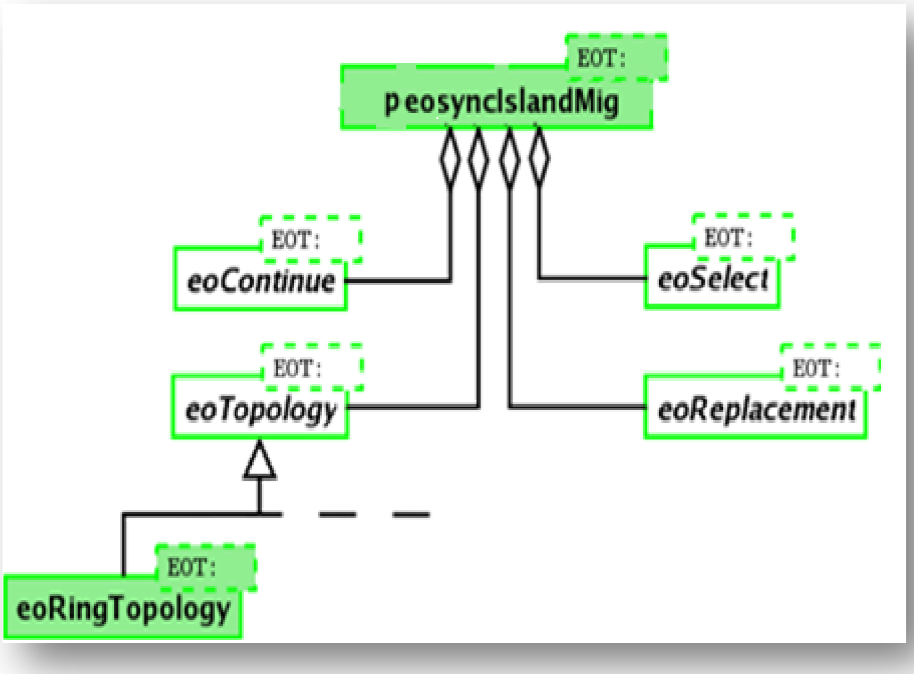


Figure 3.14. Présentation UML du modèle parallèle insulaire.

7. parallélisme de la phase transformation/évaluation :

On propose l'utilisation du modèle steady-state asynchrone motivé dans le chapite1. Le modèle steady-state asynchrone de la phase de transformation/ évaluation est présenté par la classe « eoDistAsyncEA ». La classe « eoSGTransform » englobe les opérateurs de mutation, recombinaison et évaluation. A chaque itération deux individus parmi les meilleurs individus de la population courante, sont sélectionnés et envoyés à un nœud esclave pour participer à la phase de transformation/ évaluation.

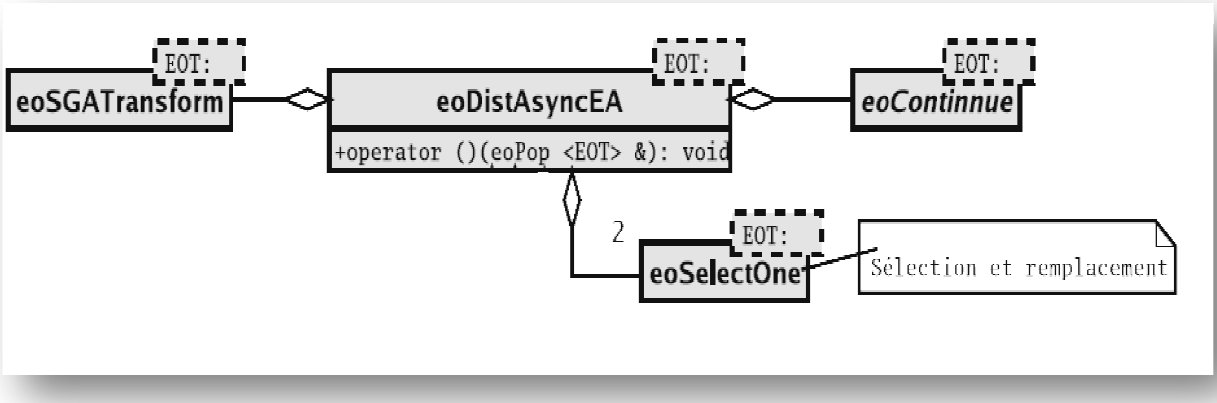


Figure 3.15. Présentation UML du modèle asynchrone de la phase de transformation/ évaluation

8. Le principe d'hybridation proposé

L'hybridation proposée pour les deux modèles parallèles est de type « LCH » (Low-level Co-evolutionary Hybrid), où l'opérateur de mutation est remplacé par une autre métaheuristique qui est le Hill Climbing.

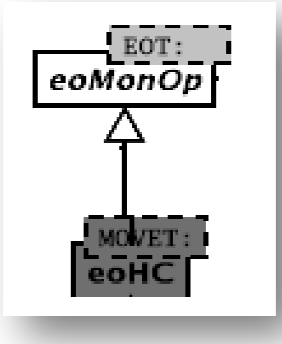


Figure 3.16. Présentation UML du modèle hybride proposée.

9. Transparence du parallélisme dans les phases de conception et de déploiement

De nombreux efforts ont été faits à la mise en œuvre pour garantir une transparence maximale de la plate-forme Paradiseo vis à vis du parallélisme et de la distribution, et ainsi faciliter son exploitation par l'utilisateur novice de ces technologies.

La mise en œuvre d'applications parallèle ou distribuées exploite une librairie pour l''´echange de messages : LAM-MPI. Nous introduisons la notion de couche de communication « eoComm ». Il s'agit d'une interface bas-niveau déclarant un ensemble de primitives nécessaires et suffisant pour l''´echange de messages et la synchronisation. Voilà la présentation UML de cette classe :

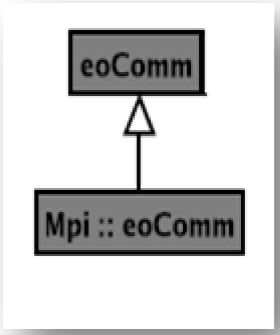


Figure 3.17. Présentation UML de la bibliothèque de communication

10. Conclusion

A travers ce chapitre, on a expliqué comment nous avons conçu nos modèles parallèles hybrides pour résoudre le problème QAP et Q3AP, cette conception est indépendante du problème dans la plus part de ces parties, et reliées dans quelques phases, la chose qui permet de l'adapter pour nombreux problème. On a profité des classes UML, pour présenter les différentes classes Paradiseo utilisées pour la conception de nos modèles, notamment les classes dédiées aux modèles parallèles et hybrides. On a choisis de paralléliser l'évolution de la population selon le modèle insulaire synchrone dans le premier modèle, et de paralléliser la fonction d'évaluation dans le deuxième modèle. L'hybridation dans les deux modèles parallèles consiste à remplacer l'opérateur de mutation de l'algorithme génétique par un algorithme Hill Climbing, dans le but de profiter des deux aspects des métaheuristiques : l'intensification, et la diversification.

CHAPITRE4

IMPLEMENTATION ET MESURE DE PERFORMANCES

1. INTRODUCTION

Dans le présent chapitre, en tente à évaluer les performances de nos modèles parallèles hybrides proposés dans le chapitre précédent, notre but était de résoudre le problème d'affectation quadratique, et d'affectation quadratique en trois dimensions.

Notre expérimentation été réalisée sous la plateforme Paradiseo, et le langage utilisé est le langage C++, malheureusement on n'a pas pu avoir des résultats pour le Q3AP, pour raison de manque de plusieurs classes nécessaires, et l'impossibilité de les concevoir dans un temps significativement réduit. En revanche, on a pu avoir des résultats pour la résolution du QAP. Les trois modèles : algorithme génétique classique, modèle insulaire parallèle hybride, et modèle de transformation/ évaluation parallèle hybride sont comparés dans ce chapitre. A chaque exécution on a une valeur de la fonction objective ainsi que la permutation qui a permet d'atteindre cette valeur.

Les performances des trois modèles sans mesurées en fonction de plusieurs paramètres : le nombre d'itération de l'algorithme génétique, le fichier d'instance (taille du problème QAP), la probabilité d'effectuer l'opérateur de mutation et de recombinaison, et le nombre d'individus immigré. Les résultats sont renforcés par des explications et des comparaisons qui mettent l'accent sur le comportement du processus d'optimisation.

2. Implémentation de l'algorithme génétique classique

L'algorithme génétique pour la résolution du QAP est déjà implémenté sous Paradiseo, on a juste précisé les paramètres suivants (Figure 4.1) :

- La taille de la population initiale est fixée à 10 et le nombre d'itération est fixé à 100.
- Le nombre des individus participant à une sélection par tournois qui est fixé à 2.
- Le fichier d'instances utilisées.
- La cartographie de communication entre différents processus définie par un fichier XML appelé schema.xml.
- Le critère d'arrêt de l'AG qui est le nombre d'itérations fixé à 100.
- Et le taux de mutation égale à 0.8 et le taux de recombinaison égale à 0.7 : les taux sont élèves dans le but d'augmenter la diversification, et la change d'atteindre des régions prometteuses.

Ces paramètres sont présentés dans le fichier Param par les lignes suivantes :

```
--popSize=10                    # -P: Population Size

--tSize=2                       #tournament size
--inst=../benchs/tai50a.dat     # -instance file
--schema=schema.xml             # -xml file
--maxGen=100                    # -G: Maximum number of generations
--pCross=0.7                    # -C: Probability of Crossover
--pMut=0.8                      # -M: Probability of Mutation
```

Les lignes de codes constituants le fichier main du programme de ce modèle sont les suivantes :

```
      // Initialise the population
 eoPop<Problem> pop(param.popSize, chromInit);
    // The robust tournament selection
eoDetTournamentSelect<Problem> selectOne(param.tSize);
    // is now encapsulated in a eoSelectPerc (entage)
 eoSelectPerc<Problem> select(selectOne);
   // CROSSOVER
 ProblemXover Xover;
   // MUTATION
 ProblemSwapMutation  mutationSwap;
   // The operators are encapsulated into an eoTRansform object
eoSGATransform<Problem>   transform(Xover,   param.pCross,   mutationSwap,
param.pMut);
   // REPLACE
```

```
eoPlusReplacement<Problem> replace;
   // generation continuation
eoGenContinue<Problem> genCont(param.maxGen);
eoEasyEA<Problem> gga(genCont, plainEval, select, transform, replace);
```

Pour exécuter ce programme, on a utilisé la ligne de commande Unix suivante :

. / exécutable ../ [instance file .dat]

La fenêtre de la figure 4.1 constitue le résultat de l'exécution du programme de l'AG classique pour le fichier d'instance Tai50a:

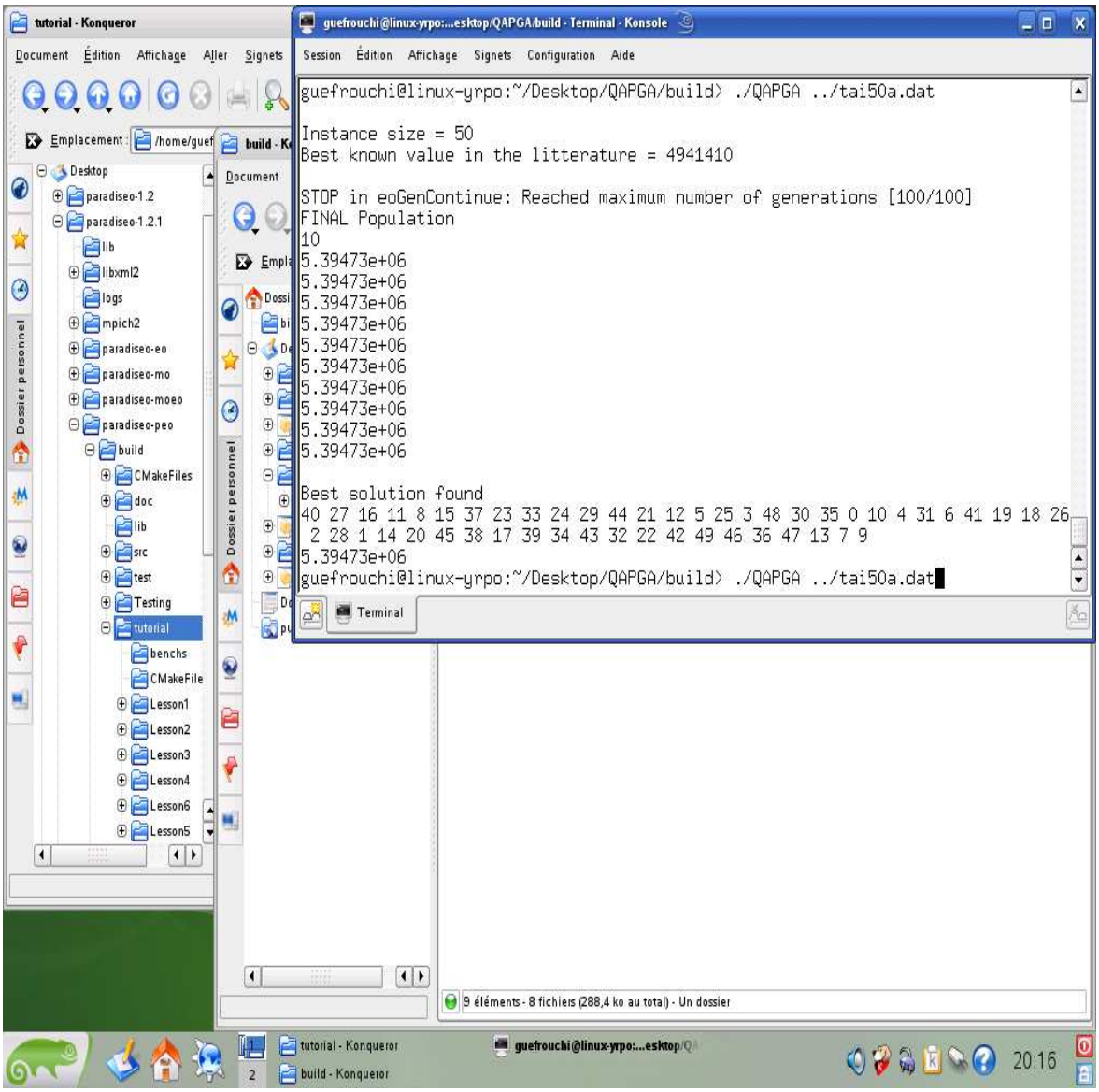


Figure 4.1. Fin d'exécution du programme (AG classique).

3. Implémentation et exécution des modèles parallèles hybrides

Avant de commencer l'implémentation de tout modèle parallèle, il faut noter que certains étapes sans indispensables :

1- Installer le package Paradiseo-PEO dédié aux métaheuristiques parallèles hybrides, disponible seulement sous UNIX.

2- Installer la bibliothèque de communication MPI.

3- Les paramètres de l'algorithme sont récolter dans un fichier nommé *PARAM*, et donc on peut tester les performances en fonction de chaque paramètre.

4- Les processus qui s'exécutent en parallèle, ont besoin de communiqué entre eux, pour cela une cartographie de communication est fournit par un fichier XML appelé schema.xml, définissant le rôle de chaque processus. Voici un exemple d'un fichier schema.xml :

```
<?xml version="1.0"?>

<schema>

   <group scheduler="0">
      <node name="0" num_workers="0">
      </node>
      <node name="1" num_workers="0">
      <runner>1</runner>
      </node>
      <node name="2" num_workers="1">
      </node>
      <node name="3" num_workers="1">
      </node>
    </group>

</schema>
```

Dans cet exemple, il ya quatre nœuds. Le premier est un nœud planificateur. Le second est le nœud sur lequel l'algorithme maître est exécuté. Le troisième et le quatrième sont des nœuds esclaves. Si on souhaite utiliser un autre nœud esclave, on ajoute dans le même fichier Xml les deux lignes suivante:

```
<node name="4" num_workers="1">
    </node>
```

Si on souhaite utiliser un autre algorithme (modèle en île), on ajoute:

```
Non-distributed
    <node name="1" num_workers="0">
      <runner>1</runner>
      <runner>2</runner>
    </node>
Distributed
    <node name="1" num_workers="0">
      <runner>1</runner>
    </node>
    <node name="2" num_workers="0">
      <runner>2</runner>
    </node>
```

Pour exécuter un programme parallèle on a des commandes avec une syntaxe bien définie :

- Lancement des démons : un démon sur un nœud de calcul gère les communications avec tous les autres nœuds. On doit avoir aux moins un démon MPD sur chaque nœud. Si on souhaite faire sur notre propre machine, cela peut se faire simplement en tapant :

```
mpd &
```

Ou bien par la commande

```
mpdboot
```

- Pour exécuter un programme d'un modèle parallèle, nous devons utiliser la fonction mpiexec. L'appel de mpiexec se fait comme suit:

```
mpiexec –n[nombre des noueds] ./ exécutable @param
```

3.1. Implémentation de l'hybridation proposée

Le fichier main contient des lignes de code qui concernent l'implémentation du modèle hybride proposé pour les deux modèle parallèle : insulaire et de la phase de transformation évaluation:

```
//mutation replaced by the hill climbing
  MoveInit move_init;
  ProblemEval problem_eval;
  MoveNext move_next;
  MoveIncrEval move_incr_eval;
  moBestImprSelect<Move> move_select;
  moHC <Move> hill_climbing (move_init, move_next, move_incr_eval,
move_select,problem_eval );

 // The operators are encapsulated into an eoTRansform object
  eoSGATransform<Problem> transform(Xover, param.pCross, hill_climbing,
param.pMut);
```

3.2. Implémentation du modèle parallèle insulaire:

Pour ce modèle il faut préciser autres paramètres, pour cela les lignes suivantes sont ajoutées dans le fichier Param:

```
######    Migration Policy     ######
--nbMigrants=10                      # -n : Number of migrants
--manGeneration=5                    # -N : Migration at N generations
```

Ces lignes correspondent aux paramètres suivants :

- La condition d'immigration qui est un nombre fixe d'itération de l'algorithme génétique fixé à 5.
- Et le nombre des éléments immigrés qui est de 10.

Les lignes de codes spécifiques à ce modèle sont les suivantes :

```
eoPop<Problem> pop_1(param.popSize, chromInit); //Initialise the population
eoPop<Problem> pop_2(param.popSize, chromInit);// Initialise the population
eoPop<Problem> pop_3(param.popSize, chromInit);// Initialise the population

  eoCheckPoint< Problem > checkpoint_1( genCont );
  eoCheckPoint< Problem > checkpoint_2( genCont );
  eoCheckPoint< Problem > checkpoint_3( genCont );

  eoEasyEA<Problem> gga_1(checkpoint_1, plainEval, select, transform,
replace);
```

```
  eoEasyEA<Problem>  gga_2(checkpoint_2,  plainEval,  select,  transform,
replace);
  eoEasyEA<Problem>  gga_3(checkpoint_3,  plainEval,  select,  transform,
replace);


      // Start the parallel EA
    if (getNodeRank()==1)
  {     apply<Problem>(eval, pop_1);
        pop_1.sort();
        cout << "Initial Population 1\n" << pop_1 << endl;
  }
     if (getNodeRank()==2)
  {     apply<Problem>(eval, pop_2);
        pop_2.sort();
        cout << "Initial Population 2\n" << pop_2 << endl;
  }
    if (getNodeRank()==3)
  {     apply<Problem>(eval, pop_3);
        pop_3.sort();
        cout << "Initial Population 3\n" << pop_3 << endl;
  }

    //Topolgy
  RingTopology ring,topology;
  eoPeriodicContinue< Problem > mig_conti_1(  param.manGeneration );
  eoContinuator<Problem> mig_cont_1(mig_conti_1,pop_1);
  eoBestSelect <Problem> mig_select_one_1;
  eoSelector       <Problem,       eoPop<Problem>       >       mig_select_1
(mig_select_one_1,param.nbMigrants,pop_1);
  eoPlusReplacement<Problem> replace_one_1;
  eoReplace <Problem, eoPop<Problem> > mig_replace_1 (replace_one_1,pop_1);
 peosyncIslandMig<  eoPop<  Problem  >,  eoPop<  Problem  >  >  mig_1
(mig_cont_1,mig_select_1,mig_replace_1,topology);
  checkpoint_1.add( mig_1 );

  eoPeriodicContinue< Problem > mig_conti_2(  param.manGeneration );
  eoContinuator<Problem> mig_cont_2(mig_conti_2,pop_2);
  eoBestSelect<Problem> mig_select_one_2;
  eoSelector       <Problem,       eoPop<Problem>       >       mig_select_2
(mig_select_one_2,param.nbMigrants,pop_2);
```

```
  eoPlusReplacement<Problem> replace_one_2;
  eoReplace <Problem, eoPop<Problem> > mig_replace_2 (replace_one_2,pop_2);
  peosyncIslandMig< eoPop< Problem >, eoPop< Problem > > mig_2
(mig_cont_2,mig_select_2,mig_replace_2,topology);
  checkpoint_2.add( mig_2 );

  eoPeriodicContinue< Problem > mig_conti_3(  param.manGeneration );
  eoContinuator<Problem> mig_cont_3(mig_conti_3,pop_3);
  eoBestSelect<Problem> mig_select_one_3;
  eoSelector <Problem, eoPop<Problem> > mig_select_3
(mig_select_one_3,param.nbMigrants,pop_3);
  eoPlusReplacement<Problem> replace_one_3;
  eoReplace <Problem, eoPop<Problem> > mig_replace_3 (replace_one_3,pop_3);
  peosyncIslandMig< eoPop< Problem >, eoPop< Problem > > mig_3
(mig_cont_3,mig_select_3,mig_replace_3,topology);
  checkpoint_3.add( mig_3 );
```

La figure 4.2 représente une capture d'écran à la fin de l'exécution du programme de ce modèle, le fichier d'instances utilisé dans ce cas est le fichier Tai50a.

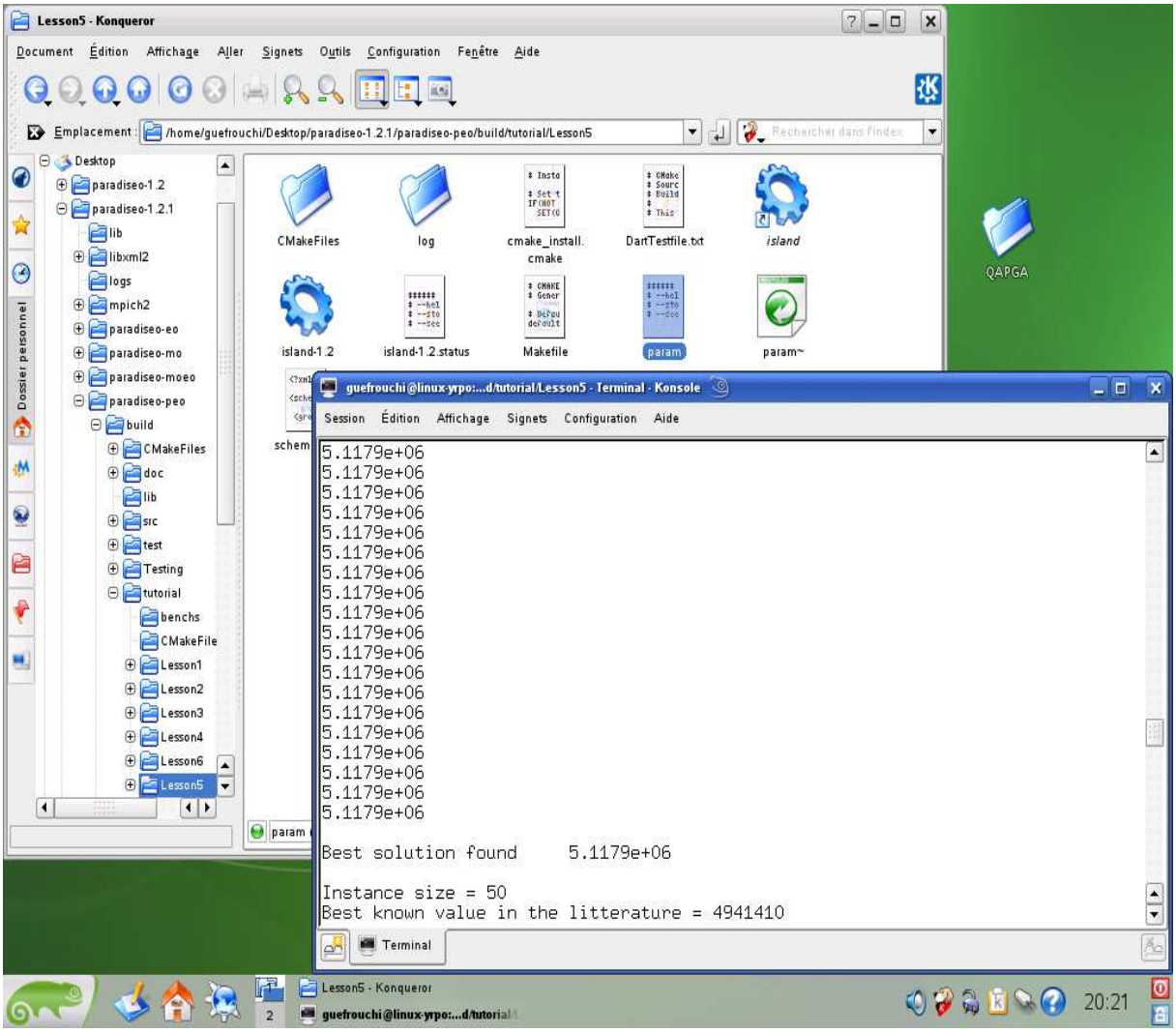


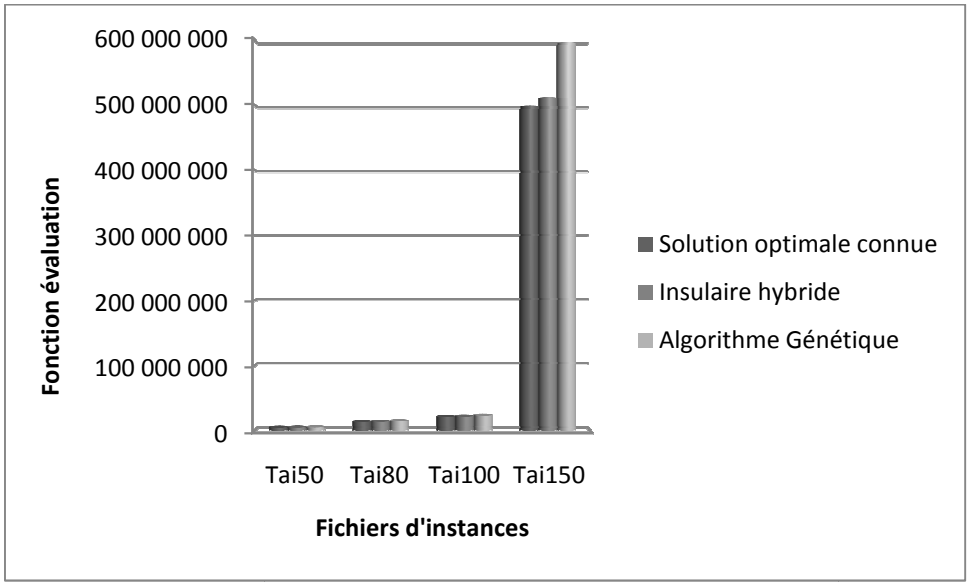
Figure 4.2. Fin d'exécution du programme (modèle insulaire parallèle hybride).

Les résultats obtenus pour chaque fichier d'instances, sont résumés dans le tableau 4.1 suivant :

Le fichier instance	Solution optimale connue	Algorithme Génétique	Insulaire hybride
Tai50	4 941 410	5 394 730	5 117 900
Tai80	13 557 864	14 790 000	13 954 000
Tai100	21 125 314	23 166 800	21 729 800
Tai150	498 896 643	595 365 000	511 458 000

Tableau 4.1. Résultats du modèle insulaire parallèle hybride.

L'histogramme dans la figure 4.1 suivant donne une interprétation des résultats du tableau précédent



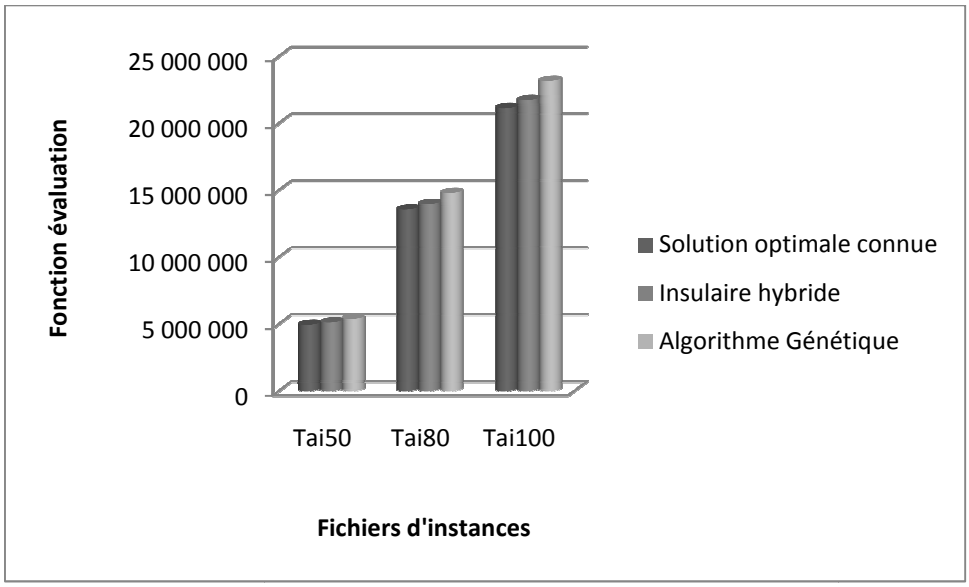


Figure 4.3. Résultats du modèle insulaire parallèle.

3.3. L'implémentation du modèle parallèle de la phase de transformation/ évaluation

Pour ce modèle il n'y a pas de paramètres spécifiques, donc le fichier Param est identique à celui du modèle de l'algorithme génétique classique. Le fichier main contient plusieurs ligne de codes on citera la plus importante, qui constitue l'utilisation de la classe eoDistAsycEA pour présenter le modèle Stenay-state asynchrone de cette phase :

```
  eoDistAsyncEA<Problem>  transform(Xover,  param.pCross,  hill_climbing,
param.pMut);
```

Voici une capture d'écran apparait après l'exécution de cet algorithme, toujours avec le fichier d'instances Tai50a :

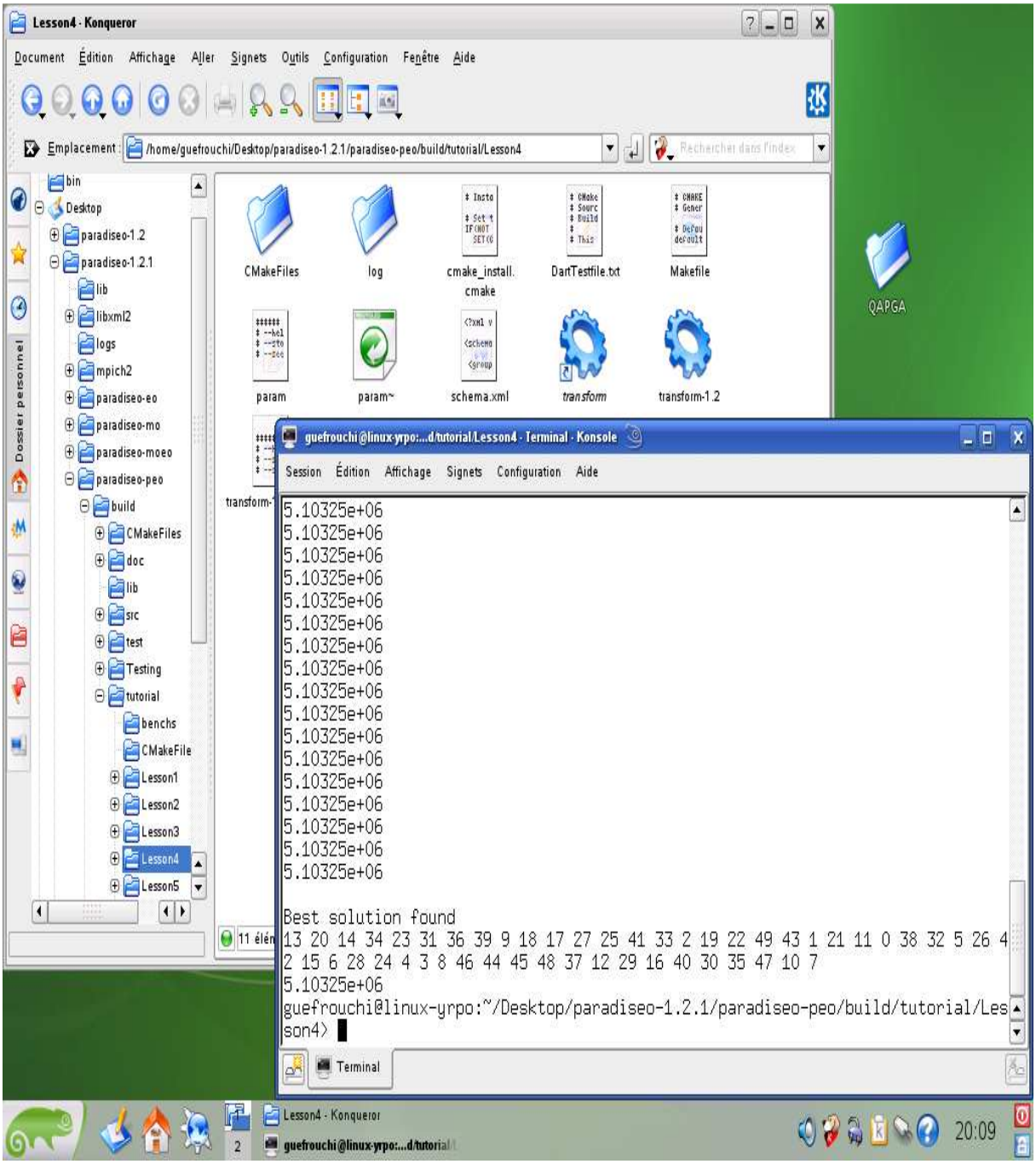


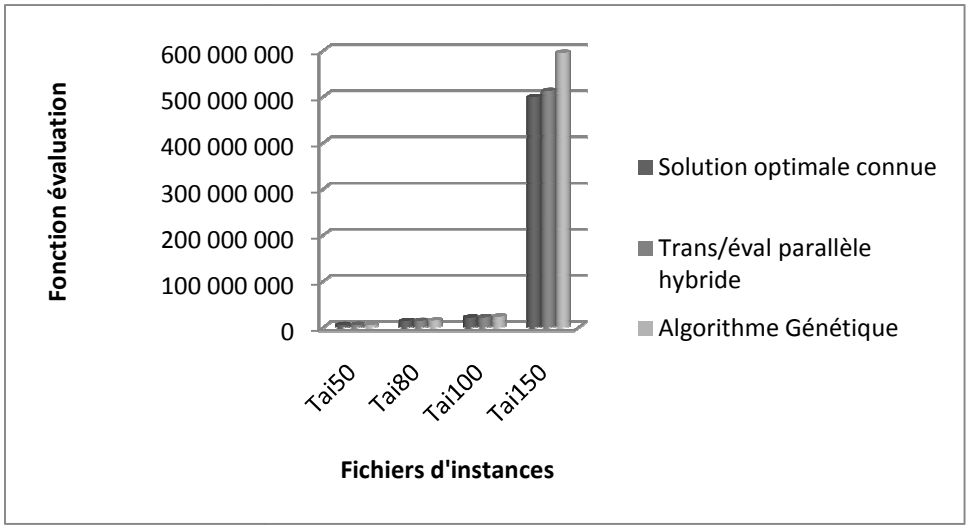
Figure 4.4. Fin d'exécution du programme (modèle parallèle hybride de la phase de transformation / évaluation).

Le tableau suivant résume les résultats obtenus pour chaque fichier d'instances:

Le fichier instance	Solution optimale connue	Algorithme Génétique	Trans/éval parallèle hybride
Tai50	4 941 410	5 394 730	5 103 250
Tai80	13 557 864	14 790 000	13 968 500
Tai100	21 125 314	23 166 800	21 599 940
Tai150	498 896 643	595 365 000	512 460 000

Tableau 4.2. Résultats du modèle parallèle hybride de transformation/évaluation.

Les résultats du tableau au dessus sont traduises en un histogramme (figure 4.6)



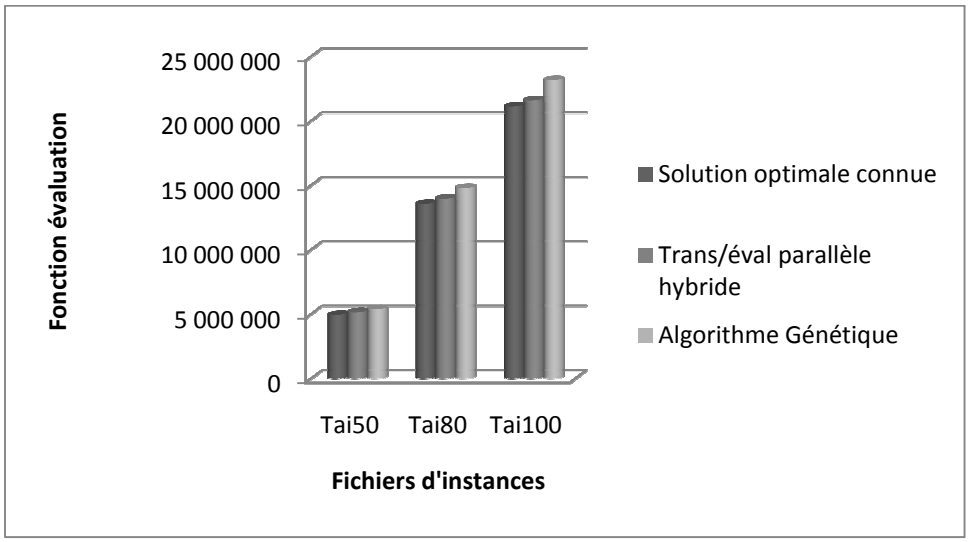


Figure 4.5. Résultats du modèle parallèle hybride de transformation/évaluation.

4. Influence de la taille de population initiale sur les résultats

A travers nos tests, on a effectué toutes les modifications possibles sur les paramètres des algorithmes (le taux de mutation, recombinaison, …), on a remarqué que deux paramètres on une influence sur la qualité de solution trouvée, il s'agit du nombre d'individus dans la population initiale générés aléatoirement, cette influence est interprétée dans le tableau, où on a choisit le fichier instance Tai50a (la valeur minimale de la fonction d'évaluation connue dans la littérature vaut 4 941 410).

Taille population initiale	**Trans/éval parallèle hybride**
10	5 103 250
50	5 095 370
80	5 085 100
100	5 055 760

Tableau 4.3. Influence de la taille de la population initiale sur les résultats.

5. Evaluation de temps d'exécution

Le temps d'exécution pour l'algorithme génétique et le modèle insulaire parallèle hybride est négligeable, mais pour le modèle de transformation/ «évaluation parallèle hybride il est distribué comme suit :

Le fichier instance	**Trans/éval parallèle hybride**
Tai50	15 seconds
Tai80	52 seconds
Tai100	2 m et 25 seconds
Tai150	15 minutes

Tableau 4.4. Influence de la taille de population initiale sur les résultats.

6. Comparaison des quatre modèles :

On remarque que l'algorithme génétique classique est un peu loin de la solution optimale par apport aux deux autres modèles. D'après le tableau.1, on remarque que tous les résultats ne sont pas forcement des solutions optimales mais sont trop proches de cette dernière. En plus on voit que le modèle insulaire parallèle hybride a donné des résultats plus performants que ceux de l'AG classique, et on a confirmé que le parallélisme de la phase de transformation/évaluation a permet de découvrir un espace de solution plus grand, ainsi on a visité des nouvelles solutions et qui sont encore plus performantes. Ces résultats sont interprétés dans l'histogramme dans la figure 4.4 suivant :

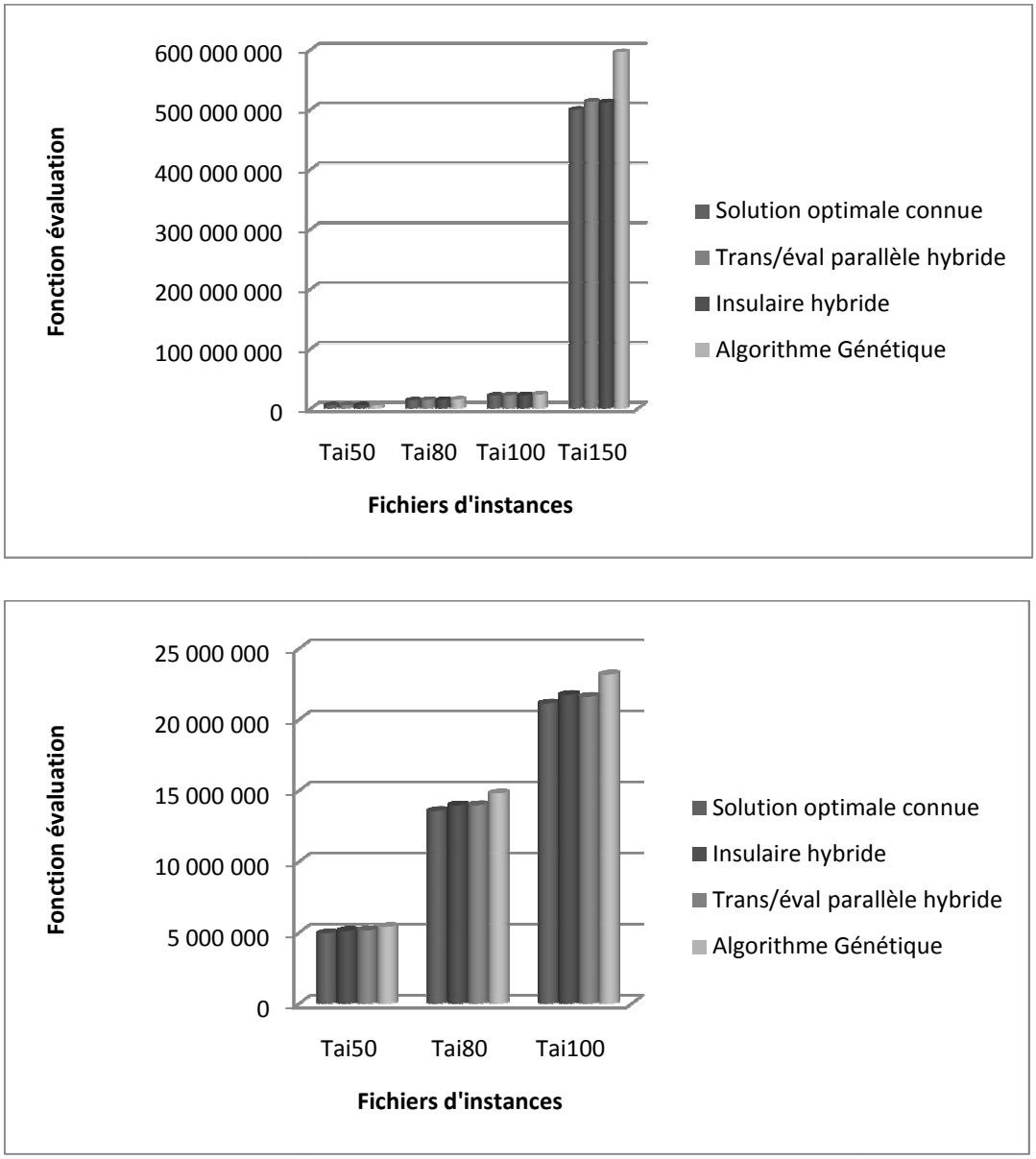


Figure 4.6. Comparaison des quatre modèles.

7. Conclusion

Ce chapitre est une récapitulation de notre expérimentation et implémentation des deux modèles parallèles hybrides : insulaire et de la phase de transformation/ évaluation. A travers cette implémentation nous avons confirmé la souplesse existante dans les étapes de conception et implémentation à l'aide de la plateforme Paradiseo. Nous avons aussi confirmé l'efficacité des métaheuristiques parallèles hybrides pour la résolution du QAP de grande taille. La séparation entre les données et les traitements, constitue une puissance et un avantage de cette plateforme. Un répertoire d'un projet sous Paradiseo contient les fichiers suivants :

- Fichier PARAM pour donner les différents paramètres des modèles.
- Fichier Main pour l'appel des classes nécessaires.
- Fichier XML pour la cartographie de communication des processus des modèles parallèles
- Les fichiers d'instances.

Conclusion Générale

Nous avons tenté à travers ce mémoire de bien comprendre les problèmes QAP et Q3AP, qui constituent des problèmes d'optimisation combinatoire très difficiles à résoudre, et qui ne possèdent pas à ce jour de solution algorithmique efficace pour les problèmes de grande taille. On a proposé une solution pour ce type de problèmes ayant un large domaine d'application pratiques. Notre proposition de solution s'articule sur les métaheuristiques les métaheuristiques et leurs aspects : intensification (exploitation), et diversification (exploration). La première est offerte par l'algorithme Hill Climbing et la deuxième par l'algorithme génétique. Ces deux notions ne sont pas contradictoires mais complémentaires et leur hybridation a permet d'améliorer la qualité de solution cherchée. Dans le but d'accélérer le processus de recherche et pouvoir supporter le coût exorbitant des algorithmes hybrides, on a utilisé les modèles parallèles qui ont montré leur supériorité.

Pour se faire notre travail à comporter les étapes suivantes :

- Adapter les algorithmes génétiques classiques pour ce type de problèmes.
- Adapter le modèle insulaire parallèle pour l'algorithme génétique classique.
- Remplacer l'opérateur de mutation de l'algorithme génétique, dans le modèle insulaire par l'algorithme Hill Climbing.
- Adapter le modèle parallèle asynchrone de la phase de transformation/ évaluation de l'algorithme génétique.
- Remplacer l'opérateur de mutation dans le modèle précédent par l'algorithme Hill Climbing.

On a implémenté nos modèle sur la plateforme Paradiseo, et on remarqué une grande souplesse et facilité dans la phase de conception. Les résultats obtenues sont pas forcement des solutions optimales mais sont trop proches de cette dernière. En plus on voit que le modèle parallèle insulaire a donné des résultats plus performants que ceux de l'AG classique, et on a confirmé que le parallélisme de la phase de transformation/évaluation a permet de découvrir un espace de solution plus grand, ainsi on a visité des nouvelles solutions et qui sont encore plus performantes.

Perspectives

Malgré les performances de nos modèles, on reste loin de découvrir une nouvelle valeur minimale. Les résultats obtenues nous a donnée l'idée de coupler les deux modèle parallèle (de transformation/évaluation et insulaire) et concevoir un nouveau modèle qu'on vise à faire fonctionner sous la plateforme Paradiseo, et que nous penserons qu'il va encore mieux guider le processus de l'optimisation, il nous reste aussi après avoir simulé le parallélisme, de déployer ces modèles sur un environnement parallèle réel comme les grilles de calculs. Il nous reste aussi de généraliser ces modèles pour le problème Q3AP et autre extensions, l'affectation quadratique générale.

ANNEXE A
UNE DESCRIPTION BREVE DE LA NOTATION UML

Dans cette annexe, nous fournissons une description brève du langage UML en nous focalisant uniquement sur les diagrammes et notations utilisées dans ce mémoire. « *Unified modeling Language* » (UML) est aujourd'hui considéré par l'organisation « *Object Management Group* » (OMG) comme la spécification standard dans la modélisation des programmes orientés objet. UML définit neuf types de diagrammes. Nous nous intéresserons uniquement ici aux diagrammes de classe.

Les diagrammes de classe décrivent la structure statique de notre plate-forme. Nous décrirons les diagrammes suivants : les classes et classes actives, la composition et l'agrégation, l'héritage, les patrons et paramètres de patron.

1. Classes et classes actives :

Ce sont les classes désignent une abstraction d'entités avec des caractéristiques communes. Elles sont représentées en Figure Annexe.1 (a) avec des rectangles divisés en compartiments. Les partitions contiennent respectivement le nom de classe (centré, en gras et en lettres capitales), la liste des attributs et celle des opérations (ou méthodes). Afin de conserver une lisibilité, seule la première partition a été considérée dans la majorité des diagrammes UML de ce mémoire. Les classes actives (Figure Annexe.1 (b)) permettent d'initialiser et de contrôler le flux d'activité, alors que les classes dites passives encapsulent des données et sont au service d'autres classes.

2. La composition et l'agrégation

Une relation d'agrégation désigne l'appartenance d'une classe *A*, dite entière (ou agrégée) et d'une classe *B*, sa partie. La class A joue un rôle important que la classe B, mais les deux classes ne sont pas dépendantes l'une de l'autre. L'agrégation est illustrée en Figure Annexe.1 (c) par un losange creux pointant vers la classe entière. La composition est une agrégation particulière dans le sens où elle dénote une appartenance forte entre les classes A et B. Elle est illustrée en Figure Annexe.1 (d) par un losange plein pointant vers la classe entière.

3. **L'héritage**.

L'héritage, désigné aussi par généralisation, est une relation « est un ». Elle est représentée en Figure Annexe.1 (e).

4. **Les patrons et modèles de patrons**.

Les patrons, nommés aussi modèles, constituent des classes génériques, qui sont représentées par des rectangles en tiret. Lorsqu'elles sont utilisées en tant que paramètres pour une classe donnée, elles sont alors représentées au coin supérieur droit de cette classe (Figure. A.1 (f)).

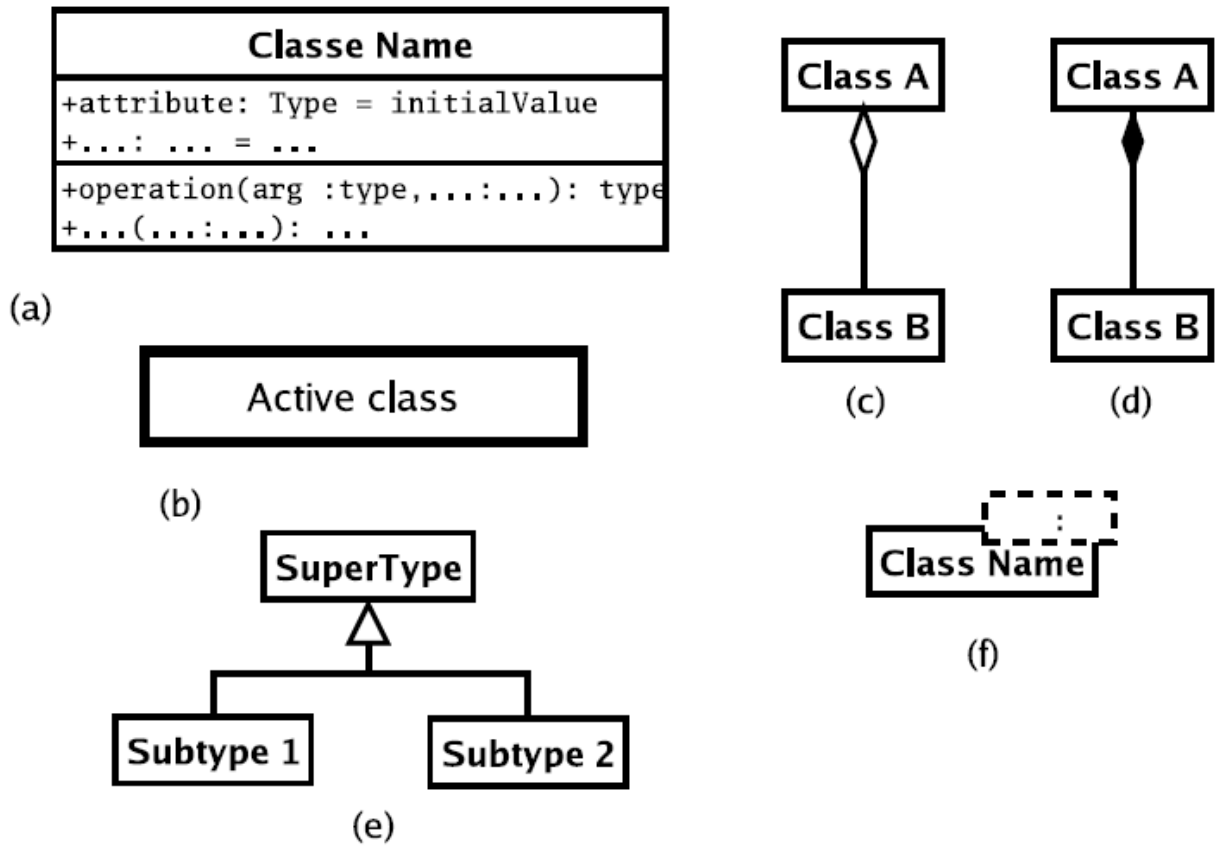


Figure Annexe.1 – Les diagrammes de classe, symboles et notations essentiels à la présentation UML.

ANNEXE B
LA MODULATION QUADRATIQUE D'AMPLETUDE

1. Généralités sur la modulation

Les ondes électromagnétiques de fréquence élevée ont la propriété de se propager, soit dans l'atmosphère, soit sur un support (guide d'onde, câble) sur de très grandes distances. Il est alors intéressant d'utiliser cette propriété afin de transmettre à distance une information. Toutefois, l'information à transmettre sera caractérisée par un signal de fréquence beaucoup plus faible. On utilise alors une "modulation". L'onde de fréquence élevée est appelée la porteuse. L'information est contenue dans la modification du signal porteur au cours du temps. Afin de transmettre l'information, il faut par conséquent :

- Fabriquer le signal contenant l'information.
- Le transmettre entre le point d'émission et celui de réception.
- Amplifier le signal reçu et éliminer le bruit
- Le démoduler pour ne retenir que l'information.

Un grand nombre de procédés de modulation numériques et analogiques sont utilisés: modulation d'amplitude, de fréquence, de phase, par impulsion, etc.

2. Principe de la modulation d'amplitude

On module l'amplitude du signal porteur selon une loi qui correspond à la forme du signal modulateur contenant l'information.

Soit : V(t) = V0 cos (Wt) la porteuse, h(t) l'information à transmettre. Le signal modulé sera donné par l'expression :

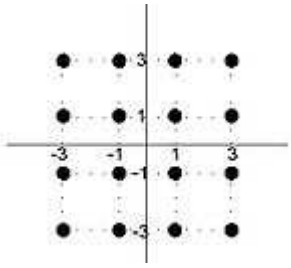
U(t) = V0 (1 + m.h(t)) cos (W**t).** Où m est appelé le taux de modulation.

3. La modulation d'amplitude en quadrature

En anglais **Quadrature Amplitude Modulation** ou **QAM**) est une forme de modulation d'une porteuse par modification de l'amplitude de la porteuse elle-même et d'une onde en *quadrature* (une onde déphasée de 90° avec la porteuse) selon l'information

transportée par deux signaux d'entrée. Autrement dit l'amplitude et la phase de la porteuse sont simultanément modifiées en fonction de l'information à transmettre.

QAM est utilisée dans les systèmes de télévision PAL et NTSC, où les signaux *en phase* et *à 90°* transportent les composantes des informations de couleur (chroma). Cette modulation est largement utilisée dans les modems, et dans d'autres formes de communications numériques sur des canaux de transport analogiques. Dans les applications numériques, le signal modulant est généralement quantifié selon ses composantes *en phase* et *à 90°*. L'ensemble des combinaisons d'amplitudes, vue sur un diagramme en (x, y), est un ensemble de points appelé *constellation QAM*. Cette constellation, et en conséquence le nombre de bits pouvant être transmis en une fois, peut être augmentée pour un meilleur débit binaire, ou diminuée pour améliorer la fiabilité de la transmission en générant moins d'erreurs binaires. Le nombre de *points* de la constellation est indiqué avant le type de modulation *QAM*. C'est un nombre entier, puissance de deux : de 2^1 (2QAM) à 2^{12} (4096QAM). La modulation 256QAM est fréquemment utilisée pour la télévision numérique par câble et dans le modem câble.



Constellation QAM à 16 états

BILBIOGRAPHIE

[1] David Hernandez : « Stratégies d'optimisation combinatoire pour le problème de l'alignement local multiple sans indels et application aux séquences protéiques ». Page 15. Université de Genève, 2005.

[2] HERVÉ MEUNIER : « Algorithmes Evolutionnaires Parallèles Pour l'Optimisation Multi-objectif de Réseaux de Télécommunications Mobiles ». Page 9. Université des sciences et technologies de Lille, le 12 Juin 2002

[3] Clarisse Dhaenens-Filipo : « Optimisation Combinatoire Multi-objectif : Apport des Méthodes Coopératives et Contribution A l'Extraction de Connaissances ». Page 40. Université des sciences et technologies de Lille, le 05 Octobre 2005.

[4] Abdelaziz Djerrah : « Résolution Exacte d'un Problème d'Optimisation Combinatoire Difficile sur Grilles de calcul».

[5] Vašek Chvátal : « Méthodes hybrides et règles de branchement en optimisation combinatoire ». Septembre 2006.

[6] Michel Van Caneghem: « Le voyageur de commerce, Algorithme "branch and bound", Algorithme Glouton, méthode de recherche locale ». Novembre 2007.

[7] Thomas Bäck, David B. Fogel, Zbigniew Michalewicz: « Evolutionary computation: advanced algorithms and operators ».2000.

[`8] Principes d'implémentation des métaheuristiques Éric D. Taillard.

[9] Sébastien Cahon : « ParadisEO : Une plate-forme pour la conception et le déploiement de métaheuristiques parallèles hybrides sur clusters et grilles ». Université des sciences et technologies de Lill, le 01/07/2005.

[10] S. Cahon, N. Melab et E-G. Talbi : « Une plate-forme pour l'optimisation parallèle approchée sur environnements de Métacomputing ». Université des sciences et technologies de Lille, le 05/04/2005.

[11] Jin-Kao Hao, Philippe Galinier, Michel Habib : « Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes ». Revue d'Intelligence Artificielle Vol : No. 1999.

[12] Daniel Costa : « Méthodes de résolution constructives, séquentielles et évolutives pour des problèmes d'affectation sous contraintes ». Lausanne, 1995.

[13] Baptiste Autin : « Les métaheuristiques en optimisation combinatoire ». Le 9 mai 2006

.

[14] Ferro Luca, Khin Samith, et Salman Nader : « Résolution pratique de problèmes NP-complets ». Page 27. Le 26 Juin 2005.

[15] Costanzo Andrea, Luong Thé Van, et Marill Guillaume : « Optimisation par colonies de fourmis ».Page 4. Le 19 mai 2006.

[16] Robin Gras : « Structure des espaces de recherche, complexité des algorithmes d'optimisation combinatoire stochastique et applications à la bioinformatique ». Page 35. Université de Rennes I, le 03 décembre 2004.

[17] Nicolas Monmarch : « Optimisation Combinatoire et Colonies de Fourmis ». Avril 1999.

[18] Mme Drdi Leïla : « Les Algorithmes Génétiques, extrait de thèse de doctorat ». 2005.

[19] Thomas Vallée Et Murat Yildizoglu : « Présentation des algorithmes génétiques et de leurs applications en économie ». Le 07 Septembre 2001.

[20] Laurent Simon : « Algorithmes Génétiques ». Université Orsay Paris 11, 2006.

[21] Philippe Muller : « Optimisation combinatoire : méthodes approchées », 2009.

[22] Fabian Lecron, Daniel Tuyttens, Pierre Manneback :« Solving the Vehicle Routing Problem with ParadisEOusing Grid 5000 ». Académie Universitaire de Wallonie Bruxelles.

[23] Souquet Amédée, et Radet Francois-Gérard : « Algorithmes Génétiques ». 21/06/2004

[24] Lassouaoui Nadia, Hamami Latifa, et Nouali Nadia : « Les Algorithmes Génétiques Application A La Segmentation Des Images », 2004.

[25] Raphael Bolze : « Analyse et déploiement de solutions algorithmiques et logicielles pour des applications bioinformatiques à grande échelle sur la grille ». Université de Lyon, le 31 Octobre 2008.

[26] Guillaume Desmottes : « Déploiement et configuration des intergiciels européens de grilles de calcul ». Université libre de Bruxelles, 2006.

[27] Mark Baker, Rajkumar Buyya, et Domenico Laforenz: « Grids and Grid technologies for wide-area distributed computing ». 2002.

[28] Foster et C. Kesselman: **«** Globus: A metacomputing infrastructure toolkit ». Journal of Supercomputer Applications, 1997.

[29] El-Ghazeli Talbi: « A Taxonomy of Hybrid Metaheuristics ». Journal of Heuristics 2002.

[30] Equipe DOLPHIN**: «** Activity report ». 2007.

[31] Olivier Roux : « La mémoire dans les algorithmes à colonie de fourmis : applications à l'optimisation et à la programmation automatique ». Université du Littoral Cote d'Opale, le 13 décembre 2001.

[32] Koopman Beckman « Assignment problems and the location of economics activities Assignment problems and the location of economics activities », 1957.

[33] P. Ji Yongzhong et Wu Haozhao Liu: « A Solution Method for the Quadratic Assignment Problem (QAP) ». The Sixth International Symposium on Operations Research and Its Applications (ISORA'06), Xinjiang, China, Aout 2006.

[34] Peter M. Hahn a, Bum-Jin Kim, Thomas Stutzle, Sebastian Kanthak, William L. Hightower, Harvind Samra, Zhi Ding, Monique Guignard: « The Quadratic Three-dimensional Assignment Problem: Exact and Approximate Solution Methods ». European Journal of Operational Research14/11/2006.

[35] Bum-Jin Kim: « Investigate of methods for solving new classes of quadratic assignment problems (QAPs) ». Université de Pennsylvanie, 2006.

[36] Djerrah Abd el-Aziz : « Résolution du problème d'affectation quadratique sur grille de calcul ». Université des sciences et technologies de Lille, février 2003.

[37] Malika Mehdi : « Combinatorial optimization using grid computing and P2P systems» Université de Luxembourg, Septembre 2007.

[38] Monique Guignard, Peter M. Hahn, Zhi Ding, Bum-Jin Kim, Harvind Samra, Thomas Stützle, et Sebastian Kanthak: « Hybrid ARQ Symbol Mapping in Digital Wireless Communication Systems based on the Quadratic 3-dimensional Assignment Problem (Q3AP) ».

[39] Abdessamad Darim **: «** Etude de l'impact du protocole TCP Sur les performances et Capacites du systeme UMTS **–**HSDPA **»** Faculté Science Technique Marrakech**, 2007.**