République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et la Recherche Scientifique
Université Mentouri de Constantine
Faculté d'Ingénieur, Département de l'Informatique

Mémoire de Magister

Spécialité : Intelligence Artificielle et Génie Logiciels

Présentée par
MAHDI SAMIR

Titre :

Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Méta/Exacte

Dirigé par le Prf. Batouche Mohamed Chawki
Professeur à l'université Mentouri de Constantine

Jury :

Président :	Dr Kholadi Mohamed Khireddine	MC, Université Mentouri de Constantine
Rapporteurs :	Dr Saïdouni Djamel-Eddine,	MC, Université Mentouri de Constantine
Examinateurs :	Dr Chikhi Salim	MC, Université Mentouri de Constantine
	Dr Chaoui Allaoua	MC, Université Mentouri de Constantine

Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Meta/Exacte

Résumé : L'impossibilité technique de résoudre exactement les problèmes NP-difficile de grande taille et/ou des problèmes avec plus de deux objectifs, impose l'utilisation des heuristiques et en particulier les métaheuristiques (génériques). Néanmoins, les méthodes exactes peuvent être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet. Ces travaux sont généralement efficaces, car les deux méthodes coopérant ont alors des particularités bien différentes, et associent leurs avantages afin d'obtenir de bons résultats. Nous présentons, dans ce papier, un nouveau schéma de coopération méta/exacte entre une métaheuristique avancée MA|PM (Memetic Algorithm with Population Management), et un algorithme exact de type branch & bound, pour la résolution de problèmes d'optimisation combinatoire multiobjectif, basée sur le concept d'optimum Pareto. Nous avons réalisé nos expérimentations sur des benchmarks bien connus dans la littérature du problème de sac à dos multiobjectif. Les résultats obtenus mettent en évidence le bon comportement de notre méthode, et la comparaison entre MA|PM et (MA|PM)⊗(B&B) montre l'efficacité de l'approche méta/exacte.

Mots-clefs : optimisation combinatoire multiobjectif, coopération méta/exacte, branch & bound, algorithme mémétique, gestion de la population, diversification, intensification.

Multi-objective Combinatorial Optimization Using a New Schema of Meta/Exact Approach

Abstract: Technical impossibility to solve exactly NP-hard combinatorial optimization problems for large instances and/or the problems with more than two objectives imposes the use of heuristics or metaheuristics. Nevertheless, the exact methods can be useful, when subproblems can be extracted from the whole problem. Their resolution indeed makes it possible to contribute to the search for the global solution, either by combining judiciously different subproblems, or by combining exact resolution of subproblems and heuristic resolution of the complete problem. This approach is generally efficient, because it combines the advantages of two very different methods. In this paper we propose to hybridize the metaheuristic MA|PM (Memetic Algorithm with Population Management) with Branch & Bound algorithm to solve combinatorial multi-objective optimization problems based on the concept of Pareto optimum. We have combined methods with the capacity of intensification and methods which have faculties of diversification, in order to obtain a good compromise between convergence towards the Pareto frontier and the distribution of the solutions along the Pareto frontier. We have realized experiments on well-known benchmarks in the literature of the multi-objective knapsack problem. The results obtained show the good behavior of our method, and the comparison between (MA|PM) and our method shows the effectiveness of Meta/Exact hybridization.

Keywords combinatorial optimization multiobjectif, co-operation meta/exact, branch & bound, memetic algorithm, population management, diversification, intensification.

Table des matières

Conclusion

Introduction

Le but de ce mémoire est de faire une synthèse en matière d'approche méta/exacte et de proposer un nouveau schéma pour la résolution de problèmes d'optimisation combinatoires multiobjectifs. L'intérêt des problèmes d'optimisation combinatoires NP-difficile est dû au large spectre de ses applications industrielles et scientifiques. En effet, que l'on s'intéresse à l'optimisation d'un système de production, au traitement d'images, à la conception de systèmes, au design de réseaux de télécommunication ou à la bio-informatique nous pouvons être confrontés à des problèmes d'optimisation combinatoire.

On peut voir de façon intuitive, un problème d'optimisation comme un problème de recherche, qui consiste à explorer un espace contenant l'ensemble de toutes les solutions potentielles réalisables, dans le but de trouver la solution optimale, sinon la plus proche possible de l'optimum, permettant de minimiser ou maximiser une fonction dite objectif :

- maximiser les performances,
- minimiser les pertes,
- Identifier une panne à moindre coût,
- maximiser le rendement du capital tout en minimisant le risque associé,
- enfin, l'ingénieur n'a pas seulement besoin de concevoir, mais il doit le faire de manière optimale.

La plupart de ces problèmes d'optimisation appartiennent à la classe des problèmes NP-difficile : classe où il n'existe pas d'algorithme qui fournit la solution optimale en temps polynomial en fonction de la taille du problème [Basseur, 2005]. La plupart des travaux réalisés dans ce domaine étaient dédiés à l'optimisation d'un seul objectif, or la plupart des applications réelles intègrent plusieurs objectifs souvent contradictoires à optimiser simultanément. Pendant longtemps, les approches de résolution multiobjectif consistaient principalement à les transformer en problèmes mono-objectifs. Depuis quelques années, l'approche Pareto définie initialement dans des travaux en économie au 19^{éme} siècle a été utilisée dans les sciences pour l'ingénieur. Cette approche a l'avantage de traiter les problèmes multiobjectifs sans transformation, sans favoriser un objectif par rapport à un autre. Dans ce cas la solution optimale ou de bonne qualité n'est plus une solution unique (cas mono-objectif) mais un ensemble de solutions compromis entre les différents objectifs à optimiser, ce qui donne une aide précieuse aux décideurs. Tout comme pour l'optimisation mono-objectif, deux classes de méthodes de résolution pour traiter les problèmes multiobjectifs : les méthodes exactes dédiées à résoudre optimalement les petites instances et les méthodes approchées : les heuristiques et en particulier les métaheuristiques (génériques) permettant d'approximer les meilleures solutions sur les plus grandes instances. La qualité d'une métaheuristique résulte en grande partie dans l'équilibre entre deux stratégies qui s'opposent : intensification (exploitation) et diversification (exploration). Ne pas préserver cet équilibre conduit à une convergence prématurée ou à une exploration trop longue, inefficace. Une façon d'améliorer les performances d'un algorithme ou de combler certaines de ses lacunes consiste à le combiner avec une autre méthode [Talbi, 2000]. L'intérêt de ces approches coopératives est de permettre à différentes méthodes d'optimisation de combiner leurs avantages dans le but d'améliorer les performances globales afin d'obtenir de bon résultats. Actuellement les meilleurs résultats obtenus sont issus de ce type d'approche, en particulier sur les problèmes réels [Basseur, 2005]. Les coopérations étaient à l'origine essentiellement réalisées entre différentes métaheuristiques (approche méta/méta). Récemment une approche hybride, assez originale visant à combiner résolutions exactes et métaheuristiques permettant de conserver aux mieux les avantages de chacune des deux approches dans le but de fournir des résultats supérieurs aux méthodes qui les composent. En effet les approches

heuristiques peuvent également fournir des informations précieuses à la méthode exacte. Les approches exactes peuvent permettre à une approche heuristique d'affiner la recherche. Par exemple une fois une région de bonne qualité localisée rapidement par un algorithme à base de population il peut être intéressant de poursuivre la recherche en appliquant une méthode exacte sur les solutions de voisinage. Des heuristiques peuvent être utilisées afin d'accélérer l'énumération des solutions en trouvant de bonnes bornes, en offrant des solutions initiales ou en définissant des plans de coupes prometteurs, aux méthodes exactes.
Dans ce mémoire, nous avons adopté une approche Pareto pour résoudre des problèmes d'optimisation combinatoires multiobjectifs. Une méthode efficace doit établir un bon compromis entre la convergence vers la frontière Pareto et la répartition des solutions le long de la frontière Pareto. Pour répondre à ces objectifs, nous avons adopté la méthode hybride méta/exacte. L'idée principale est d'hybrider résolution exacte de sous-problèmes et résolution heuristique du problème complet, en combinant des méthodes au pouvoir d'intensification et des méthodes qui possèdent des facultés de diversification. Ainsi, notre approche comporte : une méthode de ranking qui est un facteur de convergence, une méthode de diversification basée sur le crowding, une méthode d'élitisme permettant une meilleure intensification de la recherche, et une hybridation méta/exacte réalisée entre une méthode exacte de type B&B au pouvoir de recherche absolu et la métaheuristique MA|PM [Sevaux, 2005] (algorithme mémétique avec gestion de la population). L'idée d'un algorithme mémétique est de compenser la faible vitesse de convergence d'un algorithme génétique connu pour sa bonne exploration de l'espace de recherche, par l'ajout d'une méthode de recherche locale, pour mieux exploiter les bonnes solutions trouvées. La gestion contrôlée de la diversité de la population, permet en fonction de l'ensemble des individus de la population de plus ou moins diversifier et de plus ou moins intensifier la recherche. Pour l'évaluation des performances nous avons choisi le problème académique du sac à dos sous ses différentes formes "mono-objectif", "multidimensionnel" et "multiobjectif". Nous avons réalisé nos expérimentations sur des benchmarks bien connus dans la littérature. La version "mono-objectif" a été testée sur les mêmes instances que les travaux de K.H.Han et J.H.Kim [Han, 2000] obtenus par leur algorithme génétique quantique. Dans la version "multidimensionnel", les tests ont été effectués sur un nouvel ensemble d'instances référencé par OR-library à l'adresse (hces.bus.olemiss.edu/tools.html). Enfin, notre centre d'intérêt est la version "multiobjectif" basée sur le concept d'optimum Pareto et qui a été testée sur des problèmes présentés par Zitzler à l'adresse (tik.ee.ethz.ch/~zitzler/testdata.html). Les résultats obtenus mettent en évidence le très bon comportement de notre méthode sur tous les problèmes testés. Une comparaison réalisée entre, la métaheuristique MA|PM et notre méthode[1] (la même métaheuristique combinée à un B&B), montre l'efficacité de l'approche méta/exacte.

Cette thèse est organisée en 4 chapitres :

Dans le premier chapitre nous présentons les problèmes d'optimisation en général. Ensuite nous présentons en particulier les problèmes d'optimisation multiobjectifs, les différentes approches existantes pour leur résolution, ainsi que les différentes métriques permettant d'évaluer la qualité des résultats obtenus.

Le deuxième chapitre est divisé en deux parties. La première partie s'intéresse aux méthodes génériques de résolution des problèmes d'optimisation qui ont l'avantage d'être appliquées à de nombreux problèmes. Il ne s'agit pas de décrire en profondeur chacun de ces modèles mais simplement de faire une synthèse de recensement des méthodes les plus répondues en précisant leurs principales caractéristiques. La deuxième partie présente les principales métaheuristiques pour l'optimisation multiobjectif.

[1]La méthode a donné lieu à une présentation à META2006 et une publication à ISPS2007 (la version multiobjectif)

Le troisième chapitre, présente un état de l'art de la méthode coopérative méta/exacte avec des exemples trouvés dans la littérature.

Le quatrième chapitre, décrit le schéma de coopération méta/exacte proposé dans cette thèse. La méthode développée notée (MA|PM)⊗(B&B) se propose de combiner résolution exacte de sous-problèmes et résolution heuristique du problème complet en combinant des méthodes au pouvoir d'intensification et des méthodes qui possèdent des facultés de diversification. L'algorithme principal est le MA|PM : algorithme mémétique avec gestion de la population, renforcé par un algorithme exact de type Branch & Bound. Nous Présentons également le problème académique du sac à dos sous ses différentes formes sur lesquels nous avons évalué les performances de la méthode. Nous comparons notamment notre algorithme hybride avec sa version non-hybride (MA|PM), mais aussi avec les résultats obtenus dans la littérature. Nous discutons les résultats obtenus afin de conclure sur l'intérêt de telle coopération.

Enfin, nous terminons par une conclusion générale, apport, limite et perspective.

Chapitre 1

L’optimisation multiobjectif

Dans ce chapitre, nous présentons l’optimisation combinatoire multiobjectif qui sera le cadre de travail de cette thèse. Nous introduisons les concepts fondamentaux et les principales approches de résolution.

Sommaire

1 Introduction sur les problèmes d'optimisation

De nombreux secteurs de l'industrie sont concernés par les problèmes d'optimisation combinatoire. En effet, que l'on s'intéresse à l'optimisation d'un système de production, au traitement d'images, à la conception de systèmes, au design de réseaux de télécommunication ou à la bio-informatique nous pouvons être confrontés à des problèmes d'optimisation combinatoire. Plusieurs problèmes ont été traités dans différents domaines :

- design de systèmes dans les sciences d'ingénieurs (mécanique, aéronautique, chimie, etc.) : ailes d'avions [Obayashi, 1998], moteurs d'automobiles [Fujita, 1998] ;
- ordonnancement et affectation : ordonnancement en productique [Shaw, 1996], localisation d'usines, planification de trajectoires de robots mobiles [Fujimura, 1996], etc.
- agronomie : programme de production agricole, etc.
- transport : gestion de containers [Todd, 1997], design de réseaux de transport [Friesz, 1993], tracé autoroutier, etc.
- environnement : gestion de la qualité de l'air [Loughlin, 1997], distribution de l'eau [Halhal, 1997], etc.
- télécommunications : design d'antennes [Veldhuizen, 1997], affectation de fréquences [Dahl, 1995], [Weinberg, 2001], radiotéléphonie mobile [Meunier, 2002], etc.

a) Un problème d'optimisation est défini par :

- **un espace de recherche (de décision)** : ensemble de solutions ou de configurations constitué des différentes valeurs prises par **les variables de décision**.
- une ou plusieurs **fonction(s)** dite **objectif(s),** à optimiser (minimiser ou maximiser).
- un ensemble de **contraintes** à respecter.

Dans la plupart des problèmes, l'espace d'état (décision) est fini ou dénombrable. **Les variables** du problème peuvent être de nature diverse (réelle, entier, booléenne, etc.) et exprimer des données qualitatives ou quantitatives. **La fonction objectif** représente le but à atteindre pour le décideur. **L'ensemble de contrainte** définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité et permettent en général de limiter l'espace de recherche (solutions réalisables). La résolution optimale du problème consiste à trouver le point ou un ensemble de points de l'espace de recherche qui satisfait au mieux la fonction objectif. Le résultat est appelé **valeur optimale** ou **optimum**. Néanmoins en raison de la taille des problèmes réels, la résolution optimale s'est souvent montrée impossible dans un temps raisonnable. Cette impossibilité technique impose la résolution approchée du problème, qui consiste à trouver une solution de bonne qualité (la plus proche possible de l'optimum). Il est vital pour déterminer si une solution est meilleure qu'une autre, que le problème introduise un critère de comparaison (**une relation d'ordre**). La plupart des problèmes d'optimisations appartiennent à la classe des problèmes **NP-difficile** classe où il n'existe pas d'algorithme qui fournit la solution optimale en temps polynomial en fonction de la taille du problème et le nombre d'objectifs à optimiser. Dans la littérature il existe des **problèmes académiques** utilisés comme des benchmarks : *sac à dos, les fonctions de schaffer, voyageur de commerce, Flowshop, …* et des problèmes réels (applications industrielles) : télécommunications, transport, environnement, …

b) Un problème d'optimisation est caractérisé par :

- le domaine des variables de décision : soit Continu et on parle alors de **problème continu**, soit discret et on parle donc de **problème combinatoire ;**
- la nature de la fonction objectif à optimiser : soit linéaire et on parle alors de **problème linéaire**, soit non linéaire et on parle donc de **problème non linéaire** ;
- le nombre de fonctions objectifs à optimiser : soit une fonction scalaire et on parle alors de **problème mono-objectif**, soit une fonction vectorielle et on parle donc de **problème multiobjectif ;**
- la présence ou non des contraintes : on parle de **problème sans contrainte** ou **avec contrainte**
- sa taille : **problème de petite** ou **de grande taille ;**
- l'environnement : **problème dynamique** (la fonction objectif change dans le temps).

c) Face à un problème d'optimisation :

- Elaborer un modèle (mathématiques) : l'expression de l'objectif à optimiser et les contraintes à respecter.
- Développer un algorithme de résolution.
- Evaluer la qualité des solutions produites.

Dans ce chapitre nous allons aborder le premier et le troisième point. Le deuxième point sera sujet du chapitre suivant.

2 Les problèmes d'optimisation mono-objectifs

Lorsqu'un seul objectif (critère) est donné, le problème d'optimisation est mono-objectif. Dans ce cas la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal, maximal). De manière formelle, à chaque instance d'un tel problème est associé un ensemble Ω des solutions potentielles respectant certaines contraintes et une fonction d'objectif $f : \Omega \rightarrow \Psi$ qui associe à chaque solution admissible $s \in \Omega$ une valeur $f(s)$. Résoudre l'instance (Ω, f) du problème d'optimisation consiste à trouver la solution optimale $s^* \in \Omega$ qui optimise (minimise ou maximise) la valeur de la fonction objectif f. Pour le cas de la minimisation : le but est de trouver $s^* \in \Omega$ tel que $f(s^*) \leq f(s)$ pour tout élément $s \in \Omega$. Un problème de maximisation peut être défini de manière similaire.

2.1 Variables de décision

Les variables de décision sont des quantités numériques pour les quelles des valeurs sont à choisir. Cet ensemble de n variables est appelé vecteur de décision : $(x_1, x_2, \ldots, x_n)$. Les différentes valeurs possibles prises par les variables de décision x_i constituent l'ensemble des solutions potentielles.

2.2 Espace décisionnel et espace objectif

Deux espaces Euclidiens sont considérés en optimisation :

- L'espace décisionnel, de dimension *n*, *n* étant le nombre de variables de décision. Cet espace est constitué par l'ensemble des valeurs pouvant être prise par le vecteur de décision.
- L'espace objectif : l'ensemble de définition de la fonction objectif, généralement défini dans$\Re$. La valeur dans l'espace objectif d'une solution est appelée coût, ou fitness.

2.3 Contraintes

Dans la plupart des problèmes d'optimisation, des restrictions sont imposées par les caractéristiques du problème. Ces restrictions doivent être satisfaites afin de considérer une solution acceptable. Cet ensemble de restrictions, appelées **contraintes**, décrit les dépendances entre les variables de décision et les paramètres du problème. On formule usuellement ces contraintes c_j par un ensemble d'inégalités, ou d'égalités de la forme : $c_j(x_1, x_2, \ldots, x_n) \geq 0$.

3 Optimisation multiobjectif

L'optimisation multiobjectif est un axe de recherche très important à cause de la nature multiobjectif de la plupart des problèmes réels. Les premiers travaux menés sur les problèmes multiobjectifs furent réalisés au 19^{éme} siècle sur des études en économie par Edgeworth et généralisés par Pareto.

3.1 Problèmes d'optimisation multiobjectifs

Un problème d'optimisation avec objectifs multiples peut être représenté par le programme suivant :

$$\begin{cases} optimiser\ F(S) = (f_1(S), f_2(S), \ldots, f_p(S)) \\ t.q.\ S \in \Omega \text{ et } p \geq 2 \end{cases}$$

S, étant un vecteur solution $(x_1,\ldots, x_n)$ d'un espace Ω de dimension *n*, représentant des instances des variables de décision x_i. Ω représente l'ensemble des solutions réalisables respectant un ensemble de contraintes *C* d'égalité, d'inégalité et des bornes explicite. $F(f_1, f_2,\ldots, f_p)$ est le vecteur fonction objectif à optimiser, et *p* représente le nombre d'objectifs. $\Psi = F(\Omega)$ représente les points réalisables dans l'espace objectif, $Y = (y_1,\ldots, y_p)$ avec $y_i = f_i(S)$ représente un point de l'espace objectif.

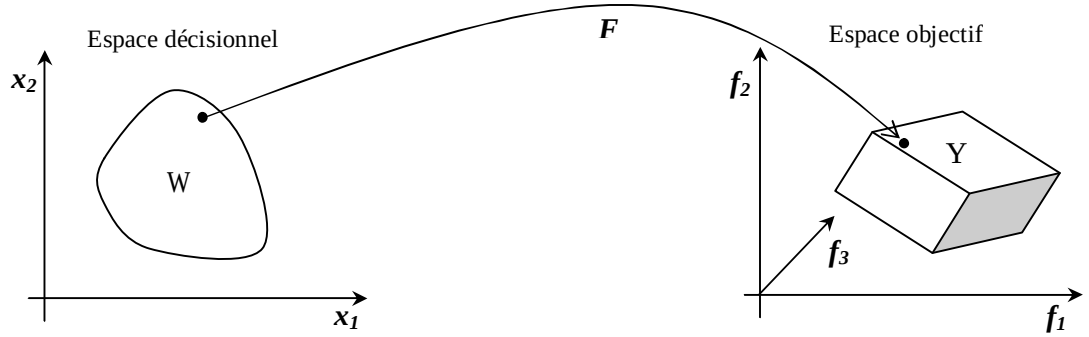


Figure 1.1 problème d'optimisation multiobjectif (2 variables de décision et 3 fonctions objectifs)

Exemple : dans le cas de deux objectifs à minimiser, toute amélioration de l'un des objectifs se fait au détriment de l'autre et que la solution optimale ou proche de l'optimum est un compromis entre les deux. Dans l'achat d'une voiture d'occasion, la voiture idéale est celle qui est peu chère (critère économique) avec peu de kilomètres (critère qualitatif), il n'est pas évident de pouvoir regrouper en un seul objectif ces deux critères non commensurables. Ainsi il n'existe plus une solution optimale unique mais un ensemble de solutions. Nous allons donc devoir identifier les meilleurs compromis possibles suivant notre budget.

Les problèmes multiobjectifs ont la particularité d'être beaucoup plus difficiles à traiter que leur équivalent mono-objectif. La difficulté réside dans l'absence d'une relation d'ordre total entre les solutions. Une solution peut être meilleure qu'une autre sur certains objectifs et moins bonne sur les autres. Donc il n'existe généralement pas une solution unique qui procure simultanément la solution optimale pour l'ensemble des objectifs. Voilà pourquoi le concept de solution optimale devient moins pertinent en optimisation multiobjectif. Dans ce cas la solution optimale ou de bonne qualité n'est plus une solution unique mais, un ensemble de solutions compromis entre les différents objectifs à optimiser. Il est vital pour identifier ces meilleurs compromis de définir une relation d'ordre entre ces éléments. La plus célèbre et la plus utilisée est la **relation de dominance** au sens **Pareto**. L'ensemble des meilleurs compromis est appelé le front Pareto, la surface de compromis ou l'ensemble des solutions efficaces. Cet ensemble de solutions constitue un équilibre, dans le sens qu'aucune amélioration ne peut être faite sur un objectif sans dégradation d'au moins un autre objectif. La solution Pareto consiste à obtenir le front de Pareto *PO* ou d'approximer la frontière de Pareto *PO**.

3.2 Approches de résolution multiobjectif

La résolution de problèmes multiobjectifs relève de deux disciplines assez différentes. En effet, résoudre un problème multiobjectif peut être divisé en deux phases :

1. **la recherche des solutions de meilleur compromis** : C'est la phase d'optimisation multiobjectif.
2. **le choix de la solution à retenir** : C'est la tâche du décideur qui, parmi l'ensemble des solutions de compromis, doit extraire celle(s) qu'il utilisera. On parle alors ici de décision multiobjectif et cela fait appel à *la théorie de la décision*.

Dans le cadre de ce mémoire nous ne parlerons que de la première phase qui consiste en la recherche des solutions de meilleurs compromis. Dans les différentes publications, nous rencontrons deux classifications différentes des approches de résolution de problèmes multiobjectifs. Le premier classement adopte **un point de vue décideur**, les approches sont classées en fonction de l'usage que l'on désire en faire. Le deuxième classement adopte **un point de vue concepteur**, les approches sont triées de leur façon de traiter les fonctions objectifs. Ainsi avant de se lancer dans la résolution d'un problème multiobjectif, il faut se poser la question du type d'approche de résolution à utiliser.

4 Classification « point de vue décideur »

On distingue à cet égard trois schémas possibles. Soit le décideur intervient des le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectif en un problème simple objectif. Soit le décideur effectue son choix dans l'ensemble des solutions proposées par le solveur multiobjectif :

- **les approches a priori :** le décideur intervient en aval du processus d'optimisation, pour définir la fonction d'agrégation modélisant le compromis que l'on désire faire entre les

différents objectifs. Dans ce cas le décideur est supposé connaître a priori le poids de chaque objectif afin de les mélanger dans une fonction unique. Cela revient à résoudre un problème mono-objectif. Cependant dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, parce que les différents objectifs sont non commensurables (exprimés dans des unités différentes).

- **les approches interactives :** combinent de manière cyclique et incrémentale les processus de décision et d'optimisation. le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi interactivement le compromis entre ses préférences et les résultats. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche.
- **les approches a posteriori** : cherche à fournir au décideur un ensemble de bonnes solutions bien réparties. Il peut ensuite, au regard de l'ensemble des solutions, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contrepartie fournir un ensemble de solutions bien réparties, ce qui peut également être difficile et requérir un temps de calcul important (mais ne nécessite pas la présence du décideur).

Nous nous placerons dans le cadre de cette troisième famille de méthodes où la modélisation des préférences n'est pas requise et où le procédé d'optimisation doit être puissant afin de fournir une très bonne approximation de la frontière Pareto.

4 Classification « point de vue concepteur »

Ce classement adopte un point de vue plus théorique articulé autour des notions d'agrégation et d'optimum Pareto. Les approches utilisées pour la résolution de problèmes multiobjctifs peuvent être classées en deux catégories [Barichard, 2003] : les approches non Pareto et les approches Pareto (*figure* 1.2).

- **Les approches non Pareto** ne traitent pas le problème comme un véritable problème multiobjectif. Elles cherchent à ramener le problème initial à un ou plusieurs problèmes mono-objectifs.
- **Les approches Pareto** ne transforment pas les objectifs du problème, ceux-ci sont traités sans aucune distinction pendant la résolution.

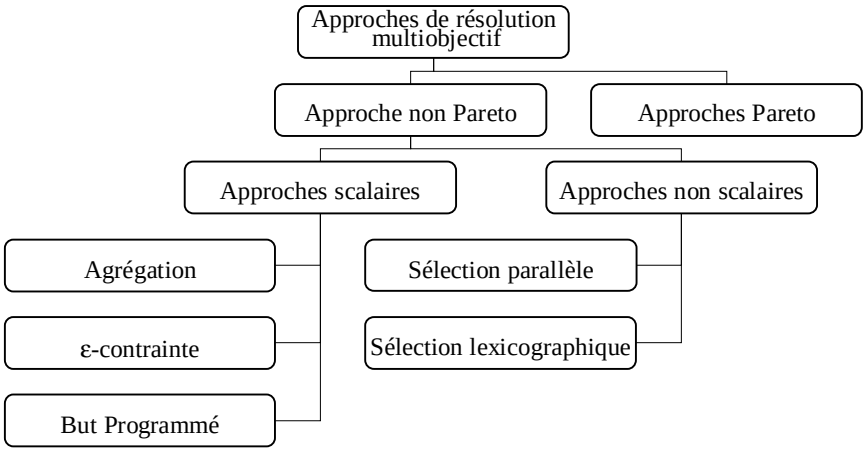


Figure 1.2 classification « point de vue concepteur »

3.2.1 Approches non Pareto

Les approches non Pareto sont classées en deux catégories : **les approches scalaires**, qui transforment le problème multiobjectif en problème mono-objectif et **les approches non scalaires**, qui gardent l'approche multiobjectif, mais en traitant séparément chacun des objectifs.

3.2.1.1 Les approches scalaires *«ces approches sont de type a priori»*

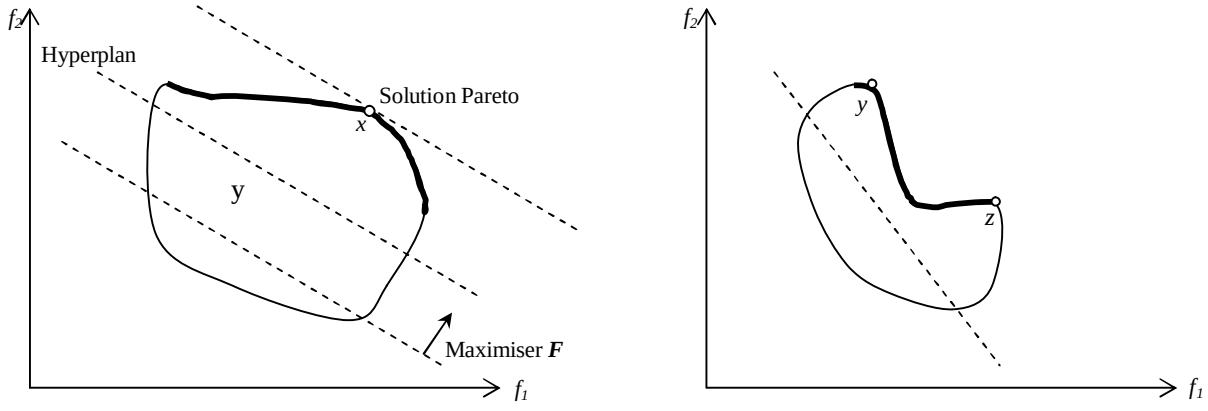
A l'origine, les problèmes multiobjectifs étaient transformés en problèmes mono-objectifs. Plusieurs approches différentes ont été mises au point pour transformer les problèmes multiobjectifs en problèmes mono-objectifs : les approches agrégées, programmation par but, et les approches ε-contraintes, etc.

a) Approche d'agrégation

C'est l'une des premières approches utilisée pour résoudre les problèmes multiobjectifs [Ishibuchi, 1998]. Elle consiste à transformer un problème multiobjectif en un problème mono-objectif, en définissant une fonction objectif unique *F* comme étant la somme pondérée des différentes fonctions objectifs du problème initial. En affectant à chaque objectif un coefficient de poids qui représente l'importance relative que le décideur attribue à l'objectif :

$$F(s) = \sum_{i=1}^{p} l_i f_i(s) \quad \text{où les poids } l_i \in [0..1] \text{ avec } \sum_{i=1}^{p} l_i = 1.$$

La *figure* 1.3 illustre le fonctionnement de la méthode d'agrégation. Fixer un vecteur poids revient à trouver un hyper-plan dans l'espace objectif (une droite pour un problème biobjectif) avec une orientation fixée. La solution Pareto optimale est le point où l'hyperplan possède une tangente commune avec l'espace réalisable (point ***x*** dans la *figure* 1.3 a). Donc, pour une agrégation donnée, il n'existe en général qu'une seule valeur optimale pour le problème. Ces approches résolvent le problème en utilisant différentes valeurs pour les poids qui fournissent différentes solutions supportées. Mais, dans le cas d'une frontière Pareto concave, les solutions non supportées sont alors négligées. La *figure* 1.3 (b) représente un cas où seules deux solutions Pareto optimales peuvent être trouvées.



(a) une frontière Pareto convexe (b) une frontière Pareto concave

Figure 1.3 l'approche d'agrégation

Cette approche a l'avantage de pouvoir réutiliser tous les algorithmes classiques dédiés aux problèmes d'optimisation à un seul objectif. Cependant cette approche a aussi deux inconvénients

importants. Le premier est dû au fait que pour avoir un ensemble de points bien repartis sur le front Pareto, les différentes valeurs l_i doivent être choisis judicieusement. Il est donc nécessaire d'avoir une bonne connaissance du problème. Le deuxième inconvénient provient du fait que cette méthode ne permet pas, de calculer intégralement la surface de compromis lorsque celle-ci n'est pas convexe.

b) But programmé

Dans les approches de ce type, le décideur doit définir des buts T_i ou références qu'il désire atteindre pour chaque objectif f_i. Ces valeurs sont introduites dans la formulation du problème, le transformant en un problème mono-objectif. La nouvelle fonction objectif est modifiée de façon à minimiser les écarts entre les résultats et les buts à atteindre.

$$\min \sum_{i=1}^{p} \left| f_i(s) - T_i \right| \text{ avec } s \in \Omega$$

Différentes approches sont envisageables, comme celles du min-max [Coello, 1998], ou du but à atteindre. Ces approches, bien que travaillant par agrégation des objectifs, permettent de générer des solutions non-supportées.

c) Approches e-contraintes

Dans cette approche, le problème consiste à optimiser une seule fonction objectif f_k sujette à des contraintes sur les autres fonctions objectif (Convertir *p-1* des *p* objectifs du problème en contraintes).

4 **Conclusion sur Les approches scalaires**

Ces différentes approches de résolution transforment un problème d'optimisation multiobjectif en un ou plusieurs problèmes à un seul objectif. Que ce soit sous la forme d'une somme pondérée, ou sous la forme d'une distance à un but, cette transformation permet d'utiliser facilement les méthodes d'optimisation issues de l'optimisation à un objectif. Mais ces méthodes ont aussi des inconvénients. Certaines ne peuvent traiter complètement des problèmes non convexes et sont donc très sensibles à la forme du front Pareto. Les autres, bien que pouvant traiter les problèmes non convexes, restent quand même sensibles à la forme du front Pareto. Un autre inconvénient important est qu'il faille relancer plusieurs fois les algorithmes de résolution avec des valeurs différentes pour certains paramètres (vecteur de poids par exemple) pour obtenir plusieurs points distincts de la surface de compromis. Ces méthodes nécessitent aussi souvent une bonne connaissance du problème a priori, notamment pour fixer les vecteurs de poids ou les points de référence. Nous présentons dans les sections suivantes comment ces méthodes surmontent les difficultés présentées précédemment.

3.2.1.2 Les approches non scalaires non Pareto *« ces approches sont de type a posteriori »*

Ces approches ne transforment pas le problème multiobjectif en un problème mono-objectif, mais utilisent des opérateurs qui traitent séparément les différents objectifs, elles n'utilisent pas non plus la notion de dominance Pareto : sélection parallèle, sélection lexicographique.

a) Sélection parallèle

Cette approche a été la première proposant un algorithme génétique pour la résolution de problèmes multiobjectifs [Schaffer, 1984]. L'algorithme proposé, VEGA (Vector Evaluated Genetic Algorithm), sélectionne les individus selon chaque objectif de manière indépendante. L'idée est simple : Pour *k* objectifs et une population de *n* individus, une sélection de *n/k*

meilleurs individus est effectuée pour chaque objectif. Ainsi *k* sous-populations vont être créées et ensuite mélangées afin d'obtenir une nouvelle population de taille *n*. le processus se termine par l'application des opérateurs génétiques (croisement et mutation).

b) Sélection lexicographique

Cette approche, proposée par Fourman [Fourman, 1985], elles classent les objectifs en fonction d'un ordre d'importance proposé par le décideur. Ensuite l'optimum est obtenu en optimisant tout d'abord la fonction objectif la plus importante puis la deuxième en intégrant les valeurs obtenues comme contraintes pour la résolution sur des objectifs moins prioritaire et ainsi de suite. La solution obtenue à l'étape *k* sera la solution du problème. Le risque essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans une niche.

4 **Conclusion sur Les approches non scalaires**

Ces différentes approches surmontent les difficultés des approches scalaires. Une seule résolution du problème permet de trouver un ensemble de solutions Pareto optimales. Le décideur peut ainsi choisir une solution suivant la situation courante. L'inconvénient de ces approches est qu'elles tendent à générer des solutions qui sont largement optimisées pour certains objectifs et très peu pour les autres. Les solutions de compromis sont négligées, et ainsi l'aspect multiobjectif du problème est contourné [Meunier, 2002].
Nous présentons dans les sections suivantes **l'approche Pareto** traitant les problèmes multi-objectifs sans transformation, sans favoriser un objectif par rapport à un autre et fournissant au décideur un ensemble compromis de solutions (supportées et non-supportées) en une seule résolution du problème.

3.2.2 Approches Pareto « ces approches sont de type a posteriori »

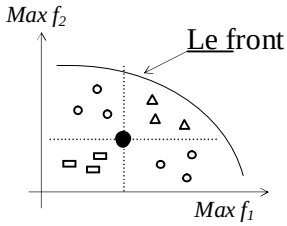
Au 19^{éme} Siècle, Vilfredo Pareto, un mathématicien Italien, formule le concept suivant : dans un problème multiobjectif, il existe un équilibre tel que l'on ne peut pas améliorer un objectif sans détériorer au moins un des autres objectifs. Les approches Pareto utilisent directement la notion de **dominance** dans la sélection des solutions générées. Le principal avantage de ces approches, c'est l'optimisation simultanée d'objectifs contradictoires.

3.2.2.1 Vocabulaire et définitions

On considère ici le cas de maximisation des objectifs. La minimisation est définie de manière analogue.

Définition 1 « dominance au sens Pareto »

Soient deux vecteurs objectifs Y^1, $Y^2 \in \Psi$ / $Y^1 = F(S^1)$ et $Y^2 = F(S^2)$. On dit que la solution S^1 **domine** S^2 (Y^1 domine Y^2) si et seulement si : $Y^1 \geq Y^2$ et $Y^1 \neq Y^2$ (ie, $y_k^1 \geq y_k^2$ pour tout k =1…p, et $y_k^1 > y_k^2$ pour au moins un k). On notera alors S^1 $\underline{\mathbf{f}}$ S^2. Si S^1 est meilleur que S^2 sur tous les objectifs (ie, $y_k^1 > y_k^2$ pour tout k =1... p) alors on dit que S^1 **domine fortement** S^2 ; On notera alors S^1 **f** S^2. Lorsque ni S^1 $\underline{\mathbf{f}}$ S^2, ni S^2 $\underline{\mathbf{f}}$ S^1, alors on dit qu'elles sont **incomparables** ou Pareto équivalentes, $S^1 \sim S^2$. La relation de dominance est une relation d'ordre partiel stricte transitive, non réflexive et non antisymétrique [Dupas 2004].



Le point noir est :

- Dominé par les triangles
- Domine les rectangles
- Incomparable aux cercles

Figure 1.4 Relation de dominance (Cas de deux objectifs à maximiser)

Définition 2

Une solution est dite Pareto optimale si elle n'est dominée par aucune autre solution réalisable.

Définition 3 « front Pareto »

L'ensemble Pareto optimal $PO^* = \{ S \in \Omega \mid \neg\exists S' \in \Omega,\ F(S') \underline{\mathbf{f}}\ F(S) \}$

L'image de l'ensemble Pareto optimal *F(PO)* dans l'espace objectif Ψ est appelée **frontière Pareto**, ou surface de compromis. L'allure de cette frontière prend des formes différentes selon que les objectifs doivent être minimisés ou maximisés, (*figure* 1.5) cas de deux objectifs.

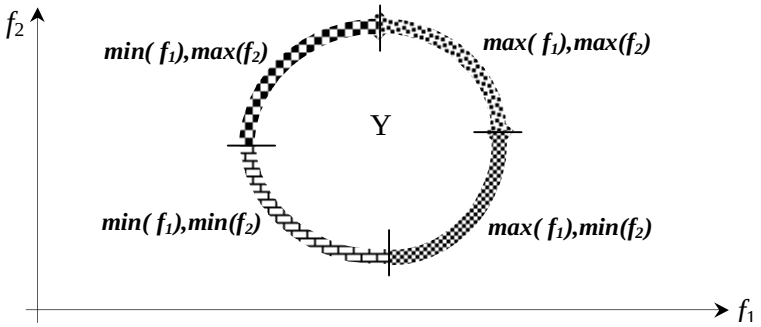


Figure 1.5 allure de la frontière Pareto selon l'optimisation (minimisation, maximisation) des différents objectifs.

L'ensemble Pareto optimal regroupe des solutions dites **supportées** correspondants aux sommets de la fermeture convexe de la frontière et des solutions **non-supportées** n'appartenant pas à cette fermeture convexe.

Définition 4 « convexité »

L'ensemble Ψ est dit convexe si tout segment joignant deux points quelconques de Ψ est inclus dans Ψ (*figure* 1.6). $y_1 \in \Psi \wedge y_2 \in \Psi \Leftrightarrow segment(y_1, y_2) \subset \Psi$

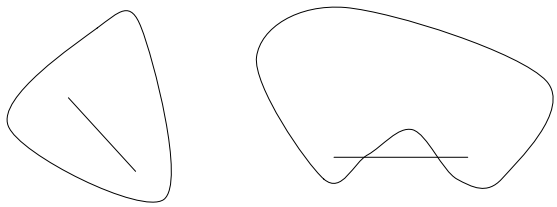


Figure 1.6 espace convexe (à gauche) et non convexe (à droite)

La convexité est le premier indicateur de la difficulté du problème. En effet, certaines méthodes sont dans l'incapacité de résoudre des problèmes non convexes de manière optimale. Mais il existe d'autres indicateurs tout aussi importants, notamment la continuité, la multimodalité, la nature des variables de décision (entières ou réelles), . . .

Définition 5 « le point idéal »

Les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des points du front Pareto. Les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objectif séparément. Dans Ψ c'est le point de cordonnées $(y_1^*,..., y_p^*)$, avec $y_k^* = \max f_k(S)$, $S \in \Omega$ et $k = 1…p$. Ce point ne correspond pas à une solution réalisable car si c'était le cas, cela sous-entendrait que les objectifs ne sont pas contradictoires et qu'une solution optimisant un objectif, optimise simultanément tous les autres, ce qui ramènerait le problème à un problème ayant une seule solution Pareto optimale. Une visualisation de l'ensemble de ces définitions est donnée sur la *figure* 1.7.

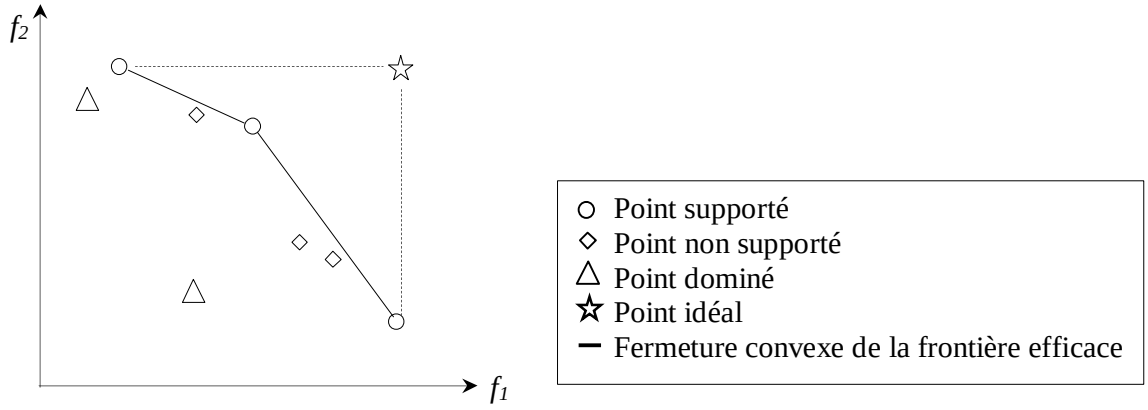


Figure 1.7 Points caractéristiques d'un problème de maximisation biobjectif

3.2.2.2 L'équilibre souhaité entre l'intensification et la diversification

Deux objectifs doivent être pris en compte dans la résolution d'un problème d'optimisation multiobjectif : **l'intensification** et **la diversification**.

– **Intensification** (exploitation) : converger vers la frontière Pareto.

– **Diversification** (exploration) : trouver les solutions diversifiées le long de la frontière Pareto :

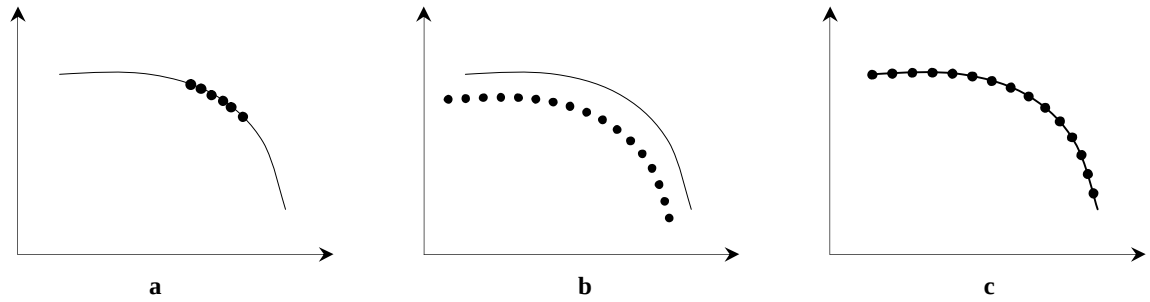


Figure 1.8 illustration de la convergence et la diversité en multiobjectif

Dans la *figure* 1.8 : (a) présente une solution de bonne qualité en terme de convergence mais mauvaise pour la diversité. (b) une solution de bonne diversité mais de mauvaise qualité pour la convergence. (c) Une disposition idéale des solutions (l'équilibre souhaité).

3.2.2.3 Mécanisme de sélection Pareto (Ranking)

L'utilisation d'un algorithme évolutionnaire dans un contexte multiobjectif nécessite de pouvoir associer une valeur scalaire unique (la fitness), au vecteur des objectifs. Ce principe appelé ranking, consiste à classer les individus en leur donnant un rang. La valeur d'adaptation est alors attribuée à chaque individu en se basant sur son rang. Cette fitness sera utilisée dans l'étape de sélection de l'algorithme (c'est ce mécanisme de sélection Pareto qui offre une alternative élégante et efficace aux algorithmes évolutionnaires de s'adapter facilement au cas multiobjectif). Plusieurs méthodes de ranking ont été utilisées dans la littérature (NSGA [Srinivas, 1995], NDS [Fonceca, 1995]). Cet ordre dépend de la notion de dominance et donc directement de l'optimalité Pareto. La méthode de ranking permet ainsi **de converger** vers les solutions Pareto optimales.

a) Ranking NSGA de Goldberg

Tous les individus non dominés de la population possèdent le rang 1. Ces individus sont ensuite enlevés de la population, et l'ensemble suivant d'individus non dominés est identifié et on leur attribue le rang 2. Ce processus est réitéré jusqu'à ce que tous les individus de la population aient un rang. Cette méthode de ranking a été utilisée dans les Algorithmes génétiques pour la résolution de plusieurs problèmes (algorithme **NSGA)**. La probabilité de sélection est ensuite affectée à chaque individu en se basant sur le rang.

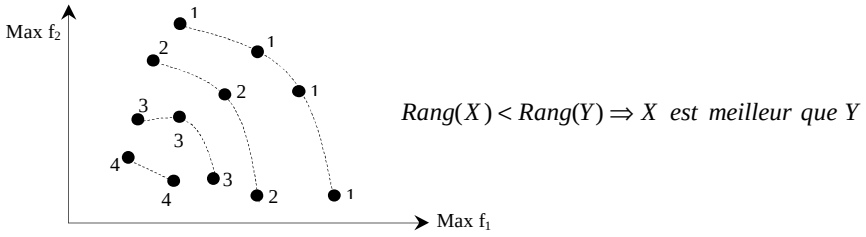


Figure 1.9 Ranking NSGA (Classement des individus par fronts)

b) Ranking NDS de Fonseca et Fleming

Dans cette méthode, le rang d'un individu est le nombre de solutions dominant l'individu plus un. Considérons par exemple un individu *i* à la génération *t*, qui est dominé par p^t_i individus dans la population courante. Son rang dans la population est donné par : $rang\ (i, t) = 1 + p^t_i$.
Un individu non dominé de la population possède donc le rang 1. Les rangs associés à cette méthode sont toujours supérieurs à ceux de la méthode NSGA. Ce type de ranking induit donc une plus forte pression de sélection, et peut causer une convergence prématurée.

3.2.2.4 L'élitisme

Consiste à maintenir une population externe, qui permet d'archiver le meilleur ensemble des points non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et participe avec une certaine probabilité à l'étape de sélection. Cette méthode permet ainsi **une intensification** de la recherche. Actuellement, les algorithmes élitistes obtiennent de meilleurs résultats sur un grand nombre de problèmes multiobjectifs [Barichard, 2003].

3.2.2.5 Méthodes de maintien de la diversité

Dans la résolution de Problèmes multiobjectifs, il est nécessaire que les solutions trouvées soient Pareto optimales, mais aussi qu'elles soient uniformément réparties dans le sous-espace des solutions Pareto optimales. Les méthodes de ranking permettent d'atteindre le premier objectif. Cependant, le deuxième objectif n'est pas pris en compte. Pour maintenir une diversité dans la

population, les méthodes de ranking doivent être utilisées en conjonction avec les techniques de maintient de la diversité. Plusieurs approches visant à maintenir la diversité dans la population ont été proposées dans la littérature : crowding, restriction de voisinage, niches écologiques (sharing). Cependant, ces techniques ajoutent un coût calculatoire non négligeable pour l'algorithme, elles doivent donc être choisies avec soin.

a) Nichage séquentiel

Dans le nichage séquentiel, la localisation de multiples niches se fait de manière séquentielle, à l'aide d'une exécution itérative de l'algorithme. Dans (Beasley et al, 1993), les auteurs ont décrit une méthode, basée sur le nichage séquentiel, pour l'optimisation de fonctions multimodales, qui évitent les inconvénients des méthodes d'exécutions itératives indépendantes. Leur stratégie est basée sur l'idée suivante : une fois qu'un optimum est trouvé, la fonction d'évaluation est modifiée dans le but de pénaliser, dans le processus de recherche, l'optimum déjà trouvé. Les étapes principales de l'algorithme sont :

1. Initialisation : affecter à la fonction coût modifiée la fonction coût originale.
2. Exécuter l'AG en utilisant la fonction coût modifiée, et en sauvegardant la meilleure solution trouvée durant la recherche.
3. Mettre à jour la fonction coût modifiée pour éviter la recherche dans les régions de la meilleure solution trouvée précédemment.
4. Si toutes les solutions n'ont pas encore été trouvées, alors retour à l'étape 2.

L'inconvénient de cette approche est qu'elle modifie la structure du problème original. D'autres méthodes avancées ont donc été proposées pour favoriser la formation de niches dans les algorithmes génétiques. Ces méthodes de diversification sont basées sur le nichage parallèle, comme par exemple les fonctions de **partage** et le **crowding** [Meunier, 2002].

b) Fonction de partage ("sharing") [Goldberg, 1987]

Le sharing consiste à modifier la valeur de coût d'un individu. Cette nouvelle valeur sera utilisée comme valeur d'adaptation par l'opérateur de sélection. Pour éviter qu'un trop grand nombre d'individus ne se concentrent autour d'un même point, il faut pénaliser la valeur d'adaptation en fonction du nombre d'individus au voisinage du regroupement : plus les individus sont regroupés, plus leur valeur d'adaptation est faible, et des individus proches les uns des autres doivent partager leur valeur d'adaptation. Dans la pratique, on estime ce taux de concentration en ouvrant un domaine autour d'un individu, puis on calcule les distances entre les individus contenus dans ce domaine. Pour déterminer les bornes du domaine ouvert autour de l'individu choisi, on définit une distance maximale, appelée s_{share}, au delà de laquelle les individus ne seront plus considérés comme faisant parti du domaine ouvert. La distance séparant deux individus i et j est calculée grâce à la fonction $d(i, j)$. La valeur d'adaptation $F(i)$ d'un individu $i \in P$ (population) est égale à son coût $F'(i)$ divisé par sa valeur de niche :

$$F(i) = \frac{F'(i)}{\sum_{j \in P} sh(d(i, j))} \text{ où la fonction } Sh \text{ est définie comme suit :}$$

$$Sh(d(i, j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{s_{share}}\right)^2 & si \quad d(i,j) \ \mathbf{p} \ s_{share} \\ 0 & \text{sinon} \end{cases}$$

La fonction *d(i,j)* de calcule de distance peut être définie dans l'espace de recherche, par exemple à l'aide d'une distance de Hamming, ou dans l'espace objectif. Ce choix dépend souvent du problème, car le maintient de la diversité dans l'espace objectif, bien qu'il soit souvent plus

simple à réaliser, n'assure pas forcément le maintient de la diversité dans l'espace de recherche [Barichard, 2003].

c) Restriction de voisinage

D'autres travaux proposés pour le maintien de la diversité dans une population sont basés sur la restriction de voisinage. L'idée est de permettre à deux individus de se reproduire s'ils sont similaires. Ceci induit la formation de différentes "espèces" (mating groups) dans la population. Dans (Loughlin and Ranjithan, 1997), les individus sont projetés sur les éléments d'une matrice de dimension *(n-1), n* étant le nombre d'objectifs. Le voisinage est restreint aux individus se trouvant dans un rayon inférieur à *r*, *r* étant fixé avant l'exécution de l'algorithme.

d) Crowding

Holland a été le premier à suggérer l'utilisation de l'opérateur de "crowding" dans la phase de remplacement des algorithmes génétiques (Holland, 1975), pour identifier les situations dans lesquelles de plus en plus d'individus dominent les niches écologiques. Dans la reproduction d'un nouvel individu, l'opérateur consiste à remplacer l'individu existant le plus semblable à l'individu généré, et non pas les parents comme dans les algorithmes génétiques standard.
La technique (crowding distance) utilisée par le NSGAII pour préserver la diversité des solutions sur le front Pareto, s'applique sur le dernier front pour compléter la taille de la population parent pour la génération suivante. L'opérateur de comparaison crowded ($<_n$) guide le processus de la sélection avec la répartition uniforme des solutions Pareto. Un individu ***i*** de la population a deux attributs : rang de non domination i_{rank} et distance crowding $i_{distance}$.
Soit deux individus *i* et *j*, $i <_n j$ si $i_{rank} < j_{rank}$ ou $(i_{rank} = j_{rank})$ et $(i_{distance} > j_{distance})$.
Avec cette relation pour la comparaison de deux solutions non dominées appartenant à deux fronts Pareto, on préfère la solution appartenant au front Pareto d'ordre le plus faible. Sinon, dans le cas où les deux solutions appartenant au même front de Pareto, on choisit la solution qui a la distance crowded la plus élevée.

3.4 Evaluation des performances

Dans tout problème d'optimisation après la modélisation et le développement d'un algorithme de résolution, vient la phase d'évaluation de la qualité des solutions produites.
La comparaison d'algorithmes pour la résolution exacte des problèmes d'optimisation est triviale. Dans le cas des méthodes approximatives, la comparaison reste triviale pour l'optimisation mono-objectif, mais pas pour l'optimisation multiobjectif. En effet l'existence d'un ensemble de solutions de compromis et l'absence d'ordre total entre les solutions rendent la mesure de qualité d'un front difficile. Si la notion de dominance au sens de Pareto peut être utilisée pour comparer deux solutions, bien que ces deux solutions puissent être incomparables, la comparaison d'un ensemble de solutions est encore plus délicate. L'évaluation d'un algorithme en terme de qualité des solutions obtenues, nécessite soit de pouvoir évaluer un front (métriques absolues) soit de le comparer de façon quantitative avec les fronts produits par d'autres algorithmes (métriques relatives). Malheureusement cette tâche est délicate puisque la notion de qualité d'un front est elle-même multiobjectif (intensification, diversification). De nombreux indicateurs de performances ont été proposés dans la littérature**.**
Dans ce qui suit *PO** représente l'ensemble des solutions potentiellement Pareto optimales trouvé par un algorithme.

3.4.1 Ensemble *PO* connu

Lorsque l'ensemble *PO* est connu, les mesures calculent le plus souvent la distance entre l'approximation *PO** et *PO*.

a) Proportion d'erreur [Veldhuizen, 2000]

Cette mesure compte le nombre de solutions *PO** qui n'appartiennent pas à *PO*, soit :

$ER = \frac{\sum_{i=1}^{|PO^*|} e_i}{|po^*|}$ où $e_i = 1$ si la $i^{ème}$ solution de $PO^* \in PO$, sinon $e_i = 0$.

Le désavantage de cette méthode est que si aucune solution de *PO** n'appartient à *PO*, elle n'apporte aucune information au sujet de la proximité relative de *PO** par rapport à *PO* puisque dans ce cas, quelque soit la distance séparant *PO** de *PO*, on *ER = 0*.

b) Distance générationnelle [Veldhuizen, 2000]

Cette mesure calcule la distance moyenne entre les solutions de *PO** et celles de *PO*. Elle se calcule selon la formule suivante : $GD = \frac{\left(\sum_{i=1}^{|PO^*|} d_i^{\,p}\right)^{\frac{1}{p}}}{|PO^*|}$

Pour $p = 2$, le paramètre d_i est la distance Euclidienne (dans l'espace des objectifs) entre la solution $i \hat{I}\ PO^*$ et le membre le plus proche de *PO*.

$d_i = \min_{k=1}^{|PO|} \sqrt{\sum_{j=1}^{p} \left(f_j^i - f_j^k\right)^2}$ où f_j^i est la valeur de la $j^{ème}$ fonction objectif de *i*, et f_j^k est la valeur de la $j^{ème}$ fonction objectif de la $k^{ème}$ solution de *PO*.

La difficulté avec cette méthode est que, s'il existe un ensemble *PO** pour lequel il y a une fluctuation importante dans la distance, la métrique peut ne pas retourner la véritable distance. Dans un tel cas, le calcul de l'écart type de la mesure est nécessaire.

c) Erreur maximale à la surface de compromis

Cette métrique permet de mesurer la distance entre *PO* et l'ensemble *PO**. Elle calcule en fait la plus grande distance minimale entre les solutions de *PO** et les solutions les plus proches de *PO*.

3.4.2 Ensemble *PO* inconnu

Lorsque l'ensemble *PO* est inconnu, les mesures permettent le plus souvent de comparer de manière relative deux approximations. Il existe cependant des mesures qui renvoient une évaluation de la qualité d'une seule approximation. Les mesures évaluent la qualité des approximations soit par rapport à la convergence, soit par rapport à la diversification, ou par rapport aux deux buts en même temps.

3.4.2.1 Mesures évaluant la convergence

a) La métrique *C*

Cette mesure, proposée par Zitzler [Zitzler, 1999], calcule la proportion de solutions d'un ensemble potentiellement Pareto optimal PO_B^* dominées par des solutions d'un ensemble PO_A^* :

$$C(A,B) = \frac{\left|\left\{ b \in PO_B^* / \exists a \in PO_A^* : a \ \mathbf{f}\ b \right\}\right|}{\left|PO_B^*\right|}$$

C(A,B) = 1 signifie que toutes les solutions trouvées par l'algorithme B sont dominées par celles trouvées par l'algorithme A. Tandis que *C(A,B)* = 0 indique qu'aucune solution générée par l'algorithme B n'est dominée par une solution trouvée par l'algorithme A. comme la relation de dominance n'est symétrique, *C(B, A)* n'est pas forcément égal à 1- *C(A,B)*. Il est donc nécessaire de calculer *C(A,B)* et *C(B, A)*.

b) Mesure de contribution [Meunier, 2000]

Cette mesurer se base également sur la comparaison de deux ensembles Pareto PO_A^* et PO_B^*. Elle permet d'avoir rapidement une idée de l'apport d'un algorithme par rapport à un autre algorithme. Soit :

- $C = PO_A^* \cap PO_B^*$,
- W_A : l'ensemble des solutions de PO_A^* qui dominent des solutions dans PO_B^*,
- L_A : l'ensemble des solutions de PO_A^* dominées par des solutions dans PO_B^*,
- N_A : l'ensemble des solutions de PO_A^* non-comparables avec toutes solutions dans PO_B^*. Alors $N_A = PO_A^* \setminus (C \cup W_A \cup L_B)$.
- W_B, L_B, N_B : idem.

- *PO** l'ensemble des solutions potentiellement Pareto optimales trouvées par les deux algorithmes *A, B* : *PO* = CÈ* W_A *È*N_A *È*W_B *È*N_B.

La contribution de l'algorithme *A* relativement à *B*, dénotée par *Contribut(A,B)*, est le ratio des solutions non dominées produites par *A* par rapport à *PO** :

$$Contribut(A,B) = \frac{\frac{|C|}{2} + |W_A| + |N_A|}{|C| + |W_A| + |N_A| + |W_B| + |N_B|}$$

Si les deux algorithmes produisent les mêmes solutions alors *Contribut(A,B)= Contribut(B,A)=1/2*
Si toutes les solutions produites par *B* sont dominées par ceux produites par *A* alors *contribut(B,A)=0*. Dans le cas général on a *Contribut(A,B)+ Contribut(B,A)= 1*.

3.4.2.2 Mesures évaluant la diversité

a) La métrique d'espacement

Cette métrique proposée par Schott [Schott, 1995], calcule la distance relative entre deux solutions consécutives de PO_A^* :

$$S = \sqrt{\frac{1}{\left|PO_A^*\right|} \sum_{i=1}^{\left|PO_A^*\right|} (d_i - \overline{d})^2} \text{ où } d_i = \min_{k \in PO_A^* \wedge k \neq i} \sum_{m=1}^{n} \left| f_m^i - f_m^k \right| \text{ et } \overline{d} = \frac{\sum_{i=1}^{\left|PO_A^*\right|} d_i}{\left|PO_A^*\right|}.$$

La distance d_i est la valeur minimale de la somme des différences absolues des valeurs des fonctions objectifs entre la $i^{ème}$ solution et toutes les autres solutions de l'ensemble généré. Il est noter que cette distance est différente de la distance Euclidienne minimale entre deux solutions. Cette métrique calcule les écarts type des différentes valeurs de d_i. Ainsi, si les solutions sont uniformément espacées, la distance correspondante sera faible. Donc, plus un algorithme trouve un ensemble de solutions pour lequel cette mesure est faible, meilleur il est.

b) Métrique maximum spread

Zitzler [Zitzler, 1999] défini une métrique mesurant la longueur de la diagonale d'une « hyper boîte » formée par les valeurs des fonctions objectifs extrêmes de l'ensemble potentiellement Pareto optimal généré : $D = \sqrt{\frac{1}{\left|PO_A^*\right|}\sum_{m=1}^{n}\left(\frac{\max_{i=1}^{\left|PO_A^*\right|} f_m^i - \min_{i=1}^{\left|PO_A^*\right|} f_m^i}{F_m^{\max} - F_m^{\min}}\right)^2}$

Où $F_m^{\max} et F_m^{\min}$ sont le maximum et le minimum pour le $m^{ème}$ objectif. Le problème de cette mesure est qu'elle ne fournit aucune information sur la distribution exacte des solutions de compromis.

c) Entropie

Cette mesure [Basseur, 2002] permet d'évaluer la diversité d'une approximation PO_A^* produite par un algorithme *A* par rapport à la diversité d'une approximation PO_B^* produite par un algorithme *B*. on notera PO^* l'ensemble des solutions non dominées de l'union de PO_A^* et PO_B^*.
Dans cette mesure, une niche est associée à chaque solution. Les solutions présentées dans chaque niche sont considérées comme voisines de la solution associée à la niche. L'entropie est alors donnée par la mesure : $E(A/B) = \frac{-1}{\log g}\sum_{i=1}^{C}\frac{1}{N_i}\frac{n_i}{C}\log\frac{n_i}{C}$

Où N_i est le nombre de solutions de $PO_A^* \cup PO^*$ se trouvant dans la niche de la $i^{ème}$ solution de $PO_A^* \cup PO^*$, C est la cardinalité de $PO_A^* \cup PO^*$, n_i est le nombre de solutions de l'ensemble PO_A^* dans la niche de la $i^{ème}$ solution de $PO_A^* \cup PO^*$, et $g = \sum_{i=1}^{C}\frac{1}{N_i}$ représente la somme des coefficients affectés à chaque solution.
Cette mesure permet donc une estimation de la diversité relative entre deux approximations. Toutefois l'interprétation des résultats n'est pas toujours évidente.

3.4.2.3 Mesures évaluant la convergence et la diversité

a) Volume de l'espace dominé par un vecteur référence

La mesure *S*, proposée par Zitzler [Zitzler, 1999], calcule l'hypervolume de la région multidimensionnelle fermé par PO_A^* et un point de référence, c'est-à-dire la taille de l'espace des objectifs que PO_A^* domine. Zitzler propose également une métrique *D* permettant de comparer deux fronts Pareto PO_A^* et PO_B^*, s'appuyant su *S*.

4 Conclusion sur les mesures d'évaluation des performances

Un ensemble de mesures ont été proposées pour analyser les performances des algorithmes multiobjectifs. Pour une bonne analyse, différentes mesures doivent être utilisées afin de pouvoir analyser à la fois convergence et diversité. L'analyse de performances en multiobjectif est en elle seule un domaine d'étude encore très ouvert puisqu'il n'existe pas de mesure universellement utilisée. Il est important de retenir qu'aucune des mesures existantes ne peut synthétiser en une seule valeur toute l'information contenue dans la surface de compromis. C'est pourquoi il est souvent nécessaire d'utiliser plusieurs de ces mesures conjointement pour espérer évaluer au

mieux la surface de compromis. Le meilleur moyen d'évaluer une surface de compromis, lorsque le problème ne comporte pas plus de trois critères, reste encore l'évaluation graphique.

4 Conclusion

Pendant longtemps, les approches de résolution multiobjectif consistaient principalement à les transformer en problèmes mono-objectifs. Parmi ces méthodes se trouvent les méthodes d'agrégation, les méthodes ε-contrainte, et les méthodes de programmation par but. La transformation du problème multiobjectif en un problème mono-objectif nécessite des connaissances a priori sur le problème traité. De plus, ces approches modifient la structure du problème combinatoire qui perd ainsi ses éventuelles propriétés remarquables. L'optimisation du problème mono-objectif peut garantir l'optimalité de la solution trouvée, mais n'en trouve qu'une seule. Dans les situations réelles, les décideurs ont généralement besoin de plusieurs alternatives. La connaissance de plusieurs solutions optimales est utile aussi pour une utilisation future, particulièrement quand la situation change et qu'une nouvelle solution est requise. Le décideur peut ainsi choisir une solution suivant la situation courante. Ces approches sont aussi sensibles au paysage de la frontière Pareto (convexité, discontinuité, etc.). Les méthodes d'agrégation linéaire, par exemple, ne trouvent que les solutions supportées (solution appartenant à l'enveloppe convexe des solutions réalisables). Depuis quelques années, l'approche Pareto définie initialement dans des travaux en économie au 19^{éme} siècle a été utilisée dans les sciences pour l'ingénieur. Cette approche a l'avantage de traiter les problèmes multiobjectifs sans transformation, sans favoriser un objectif par rapport à un autre et peuvent générer des solutions supportées et non supportées en un passage unique de l'algorithme. Elles sont basées sur la notion de dominance définie comme une relation d'ordre partiel entre les solutions du problème.

Dans le cadre de notre travail on s'intéresse aux problèmes d'optimisation combinatoire multiobjectifs. Nous avons adopté une approche de résolution Pareto qui comporte une méthode de ranking NSGA, une méthode de diversification basée sur le crowding de Holland et une méthode d'élitisme. L'évaluation des performances sera utilisée sur des instances bi-objectif dont les résultats sont représentables graphiquement.

Chapitre 2

Les méthodes de résolution

Ce chapitre est organisé en deux parties : **la première partie** présente une synthèse de recensement des méthodes d'optimisation mono et multiobjectifs les plus répondues. **La deuxième partie** présente les principales métaheuristiques pour l'optimisation multiobjectif.

Chapitre 2

Première partie

Dans cette partie du chapitre nous nous intéressons en général aux méthodes génériques de résolution des problèmes d'optimisation qui ont l'avantage d'être appliquées à de nombreux problèmes. Il ne s'agit pas de décrire en profondeur chacun de ces modèles mais simplement de faire une synthèse de recensement des méthodes les plus répondues en précisant leurs principales caractéristiques.

Sommaire

---

1 Introduction

Tout comme pour l'optimisation mono-objectif, deux classes de méthodes de résolution pour traiter les problèmes multiobjectifs : les méthodes exactes dédiées à résoudre optimalement les petites instances et les méthodes approchées : les heuristiques et en particulier les métaheuristiques (génériques) permettant d'approximer les meilleures solutions sur les plus grandes instances. Enfin, le choix de la méthode de résolution à mettre en oeuvre dépendra souvent de la complexité du problème. En effet, suivant sa complexité, le problème pourra ou non être résolu de façon optimale. Dans le cas de problèmes classés dans la classe P, un algorithme polynomial a été mis en évidence. Il suffit donc de l'utiliser. Dans le cas de problèmes NP-difficiles, deux possibilités sont offertes. Si le problème est de petite taille, alors un algorithme exact permettant de trouver la solution optimale peut être utilisé (Branch & Bound, programmation dynamique...). Malheureusement, ces algorithmes par nature énumératifs, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles (même si en pratique la taille n'est pas le seul critère limitant). Dans ce cas, il est nécessaire de faire appel à des heuristiques permettant de trouver de bonnes solutions approchées. Parmi ces heuristiques, on trouve les métaheuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes.

L'objectif est de concevoir des algorithmes de plus en plus performants
- en temps de calculs (raisonnable)
- en qualité de solutions produites (bonne qualité)
- et permettant de traiter des problèmes de plus grande taille.

4 C'est un sujet ouvert « pas de règles ». Le théorème ***« No Free Lunch »*** [ ] : pas de meilleur algorithme d'optimisation quelque soit le problème considéré (ce qu'un algorithme gagne sur certains problèmes est perdu sur d'autres).

Une façon d'améliorer les performances d'un algorithme ou de combler certaines de ses lacunes consiste à le combiner avec une autre méthode [Talbi, 2000]. L'intérêt de ces approches coopératives est de permettre à différentes méthodes d'optimisation de combiner leurs avantages dans le but d'améliorer les performances globales afin d'obtenir de bon résultats. Actuellement les meilleurs résultats obtenus sont issus de ce type d'approche, en particulier sur les problèmes réels [Basseur, 2005]. Les coopérations étaient à l'origine essentiellement réalisées entre différentes métaheuristiques. Récemment une approche hybride, assez originale visant à combiner résolutions exactes et métaheuristiques permettant de conserver au mieux les avantages de chacune des deux approches dans le but de fournir des résultats supérieurs aux méthodes qui les composent.

2 Les méthodes exactes

Les méthodes exactes cherchent à trouver de manière certaine la solution optimale en examinant de manière explicite ou implicite la totalité de l'espace de recherche. Elles ont l'avantage de garantir la solution optimale néanmoins le temps de calcul nécessaire pour atteindre cette solution peut devenir très excessif en fonction de la taille du problème (explosion combinatoire) et le nombre d'objectifs à optimiser. Ce qui limite l'utilisation de ce type de méthode aux problèmes bi-objectifs de petites tailles. Ces méthodes génériques sont : *Branch & Bound*,

Branch & Cut, Branch & Price, Branch, Cut & Price. D'autres méthodes sont moins générales, comme : *La programmation dynamique, simplexe, La programmation linéaire en nombre entiers, L'algorithme A**. D'autres méthodes sont spécifiques à un problème donné comme l'algorithme de Johnson pour l'ordonnancement.

a) Branch & Bound [ ]

Est une méthode générique de résolution exacte de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire. C'est une méthode constructive de recherche arborescente qui utilise l'énumération implicite basée sur la notion de bornes afin d'éviter l'énumération de larges classes de mauvaises solutions. Elle utilise la stratégie diviser pour régner, en se basant sur deux concepts : **le branchement** (séparation) qui consiste à partitionner ou diviser l'espace des solutions en sous problèmes pour les optimiser chacun individuellement ; et **l'évaluation** qui consiste à déterminer l'optimum de l'ensemble des solutions réalisables associé au nœud en question ou, au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution optimale, la méthode la plus générale consiste à borner le coût des solutions contenues dans l'ensemble. Les techniques les plus classiques pour le calcul de bornes sont basées sur l'idée de relaxation de certaines contraintes : relaxation continue, relaxation lagrangienne...

b) Branch & Cut

Padberg et Rinaldi [Padberg, 1991] ont amélioré l'idée du *B&B* basé sur la programmation linéaire en décrivant une méthode utilisant des inégalités renforçant la relaxation par programmation linéaire. Le principe de base est le même que pour le *B&B*, mais au lieu de brancher sur une variable pour descendre dans l'arbre de recherche, on recherche des plans de coupes qui permettent de restreindre l'espace des solutions réalisables.

c) Branch & Price

Les algorithmes de *B&B* résolvant des programmes linéaires en générant les variables dynamiquement quand cela est nécessaire sont appelés algorithmes de *Branch & Price*. Les inégalités d'un programme linéaire en nombre entier peuvent être vues comme une matrice, chaque colonne correspondant à une variable, et chaque ligne correspondant à une inégalité. Comme pour les plans cde coupes, les colonnes de la matrice correspondant aux inégalités prises en compte peuvent être définies implicitement si le nombre de variables est grand. Si une colonne n'est pas présente dans la matrice courante, alors la variable correspondante prend implicitement la valeur zéro. Le processus de générer dynamiquement les variables est appelé 'pricing'.

d) Branch, Cut & Price

Lorsque les variables et les plans de coupes sont générés dynamiquement durant l'algorithme de *B&B*, on appelle cette technique *Branch, Cut & Price*. Il existe une certaine symétrie entre la génération de coupes est de variables. Cependant, même si les méthodes de *B&C* et de *B&P* sont proches, combiner les deux approches requiert la mise en place de méthodes sophistiquées. Un aperçu des différents travaux répertoriés utilisant cette approche est consultable à l'adresse : http://branchandcut.org/reference.html.

e) Méthode à deux phases

La méthode deux-phases a initialement été proposée par Ulungu et Teghem pour la résolution d'un problème d'affectation bi-objectif [Ulungu, 1995]. Comme son nom l'indique, cette

méthode est décomposée en deux étapes : la première consiste à trouver toutes les solutions supportées du front Pareto, puis la deuxième phase cherche de façon indépendante les solutions non supportées situées entre tous les couples de solutions supportées adjacentes. Cette méthode travaille donc essentiellement dans l'espace objectif.
Durant la première phase de la méthode, les deux solutions extrêmes (solutions optimisant chacune un des deux objectifs) sont recherchées (voir *figure* 2.1.a). Puis, de façon récursive, dès que deux solutions supportées *r* et *s* sont trouvées, la méthode recherche d'éventuelles autres solutions supportées entre *r* et *s*, à l'aide de combinaisons linéaires bien choisies des objectifs (voir *figure* 2.1.b et 2.1.c). A la fin de la première phase l'ensemble des solutions supportées est donc trouvé (voir *figure* 2.1.d). Cette première phase rappelle la méthode dichotomique.
La deuxième phase consiste alors en la recherche des solutions non supportées appartenant au front Pareto. Ces solutions ne peuvent être obtenues par combinaisons d'objectifs. Ulungu et Teghem proposent alors d'utiliser les solutions supportées trouvées pour réduire l'espace de recherche en argumentant que les solutions Pareto non supportées restantes sont forcément dans les triangles rectangles basés sur deux solutions supportées consécutives (voir *figure* 2.1.e). Ainsi, une recherche de type deuxième phase est exécutée entre chaque couple de solutions supportées adjacentes (voir *figure* 2.1.f et 2.1.g). La méthode de recherche au sein de ces triangles dépend du problème étudié. A la fin de la deuxième phase, toutes les solutions Pareto sont trouvées.

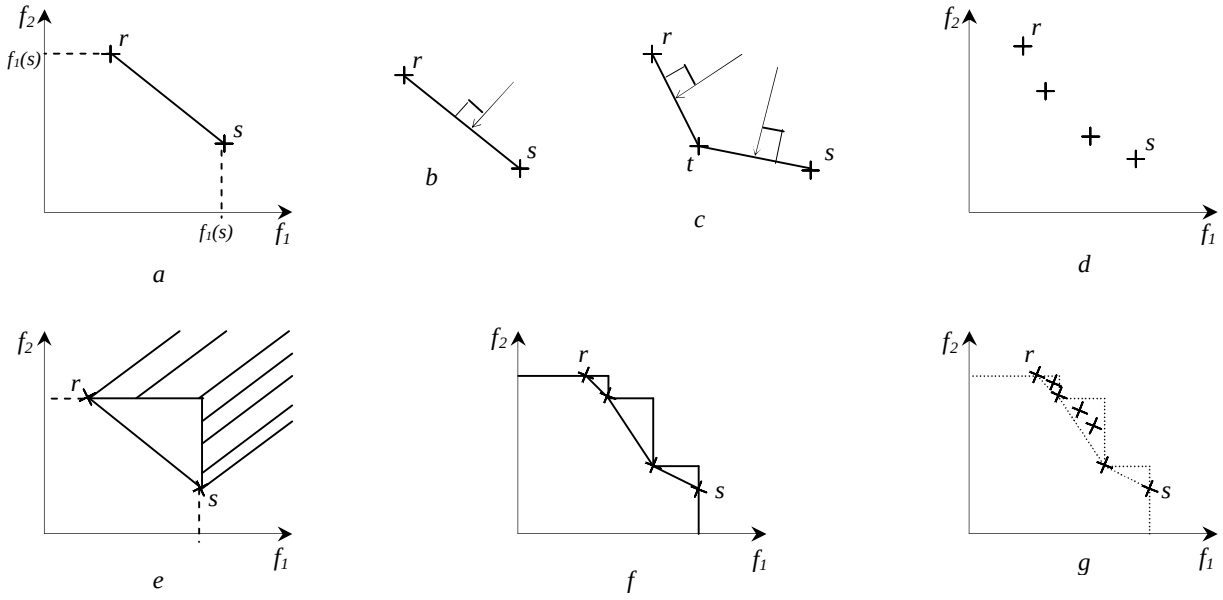


Figure 2.1 Illustration des différentes étapes de la méthode deux phases.

3 Les méthodes approchés

Méthodes souvent inspirées de mécanismes d'optimisation rencontrés dans la nature. Elles sont utilisées pour les problèmes où on ne connaît pas d'algorithmes de résolution en temps polynomial et pour lesquels on espère trouver une solution approchée de l'optimum global. Elles cherchent à produire une solution de meilleure qualité possible dictée par des heuristiques avec un temps de calcul raisonnable en examinant seulement une partie de l'espace de recherche. Dans ce cas l'optimalité de la solution n'est pas garanti ni l'écart avec la valeur optimal. Parmi

ces heuristiques, on trouve les métaheuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes.
Plusieurs classifications des métaheuristiques ont été proposées, la plupart distinguent globalement deux catégories : celles se basant sur **une solution unique** et celles se basant sur **une population de solution**.

3.1 Métaheuristiques à base de solution unique

Travaillent sur un seul point de l'espace de recherche à un instant donné en commençant avec une solution initiale puis de l'améliorer itérativement en choisissant une nouvelle solution dans son *voisinage*.
Voisinage de taille. $e : N(X') = \{X \in S / \|X - X'\| < e\}$, Minimum local : $\forall X \in N(X')$ on a $F(X') \pounds F(X)$.
On définit la notion d'optimalité locale au sens Pareto : un sous-ensemble de solutions non dominées dans un voisinage déterminé.

3.1.1 Algorithmes de recherche locale pour une optimisation locale

Tout algorithme basé sur la notion de voisinage, piégé par le premier optimum local. Le principe est simple, à partir d'une configuration initiale X_0 on applique successivement des transformations à la solution courante (mouvement). La recherche consiste à passer d'une solution à une solution voisine. De manière générale, les opérateurs de recherche locale s'arrêtent quand une solution localement optimale est trouvée, c'est à dire quand il n'existe pas de meilleure solution dans le voisinage.

a) Les algorithmes de descente

- Descente simple « simple descent »

Le principe est simple, à partir d'une solution existante, chercher aléatoirement une solution dans le voisinage et accepter cette solution si elle améliore la solution courante.

- Plus grande descente « Deepest descent »

Est une version plus agressive de l'algorithme de descente. Au lieu de choisir une solution X' dans le voisinage de X, on choisit toujours la meilleure solution X' du voisinage de X. et on se trouve très rapidement bloqué dans un optimum local.

- Multi-start descent

Dans les deux algorithmes précédents, l'équilibre souhaité entre intensification et diversification n'existe donc plus. Un moyen très simple de diversifier la recherche peut consister à réexécuter un des algorithmes en prenant un autre point de départ. Comme l'exécution de ces algorithmes est souvent très rapide, on peut alors inclure cette répétition au sein d'une boucle générale. On obtient alors un algorithme de type « Multi-start descent ».

Dans ces algorithmes, il est clair que si la fonction comporte beaucoup de minima locaux, on est certain de rester bloqué sur l'un d'eux. Il serait alors intéressant de pouvoir s'échapper de ces minima locaux : la stratégie consiste à favoriser les descentes, mais sans interdire tout à fait les remontées.

3.1.2 Algorithmes de recherche locale pour une optimisation globale

Tout algorithme local par son mécanisme de fonctionnement et global par sa recherche d'optimum global utilisant des mécanismes pour s'échapper de ces minima locaux.

a) Recuit simulé « Simulated annealing »

Le recuit simulé a été Introduit par Kirkpatrick en 1982. Il tire son origine de la mécanique statistique en utilisant les critères de la méthode de Métropolis 1953. Son principe de fonctionnement repose sur une imitation du phénomène de recuit en science des matériaux basé sur les principes d'équilibre énergétique lors de la cristallisation des métaux.
L'analogie avec une méthode d'optimisation est trouvée en associant une solution à un état du métal, on équilibre thermodynamique est la valeur de la fonction objectif de cette solution. Passer d'un état du métal à un autre correspond à passer d'une solution à une solution voisine. Le recuit simulé est une technique stochastique de type **Monte-Carlo** généralisé pour la résolution approchée de problèmes NP-difficile de l'optimisation combinatoire basé sur un paramètre appelé température, qui sera ajusté au cours de la recherche. A partir d'une solution initiale il recherche dans son voisinage une autre solution de façon aléatoire qui peut être de moins bonne qualité permettant d'échapper aux optima locaux en acceptant temporairement une dégradation de la fonction objectif. Initialement, la température est élevée autorisant une forte dégradation de qualité puis décroît pour diminuer la probabilité des dégradations importante.

b) Recherche tabou « Tabu search »

Le principe de base est de poursuivre la recherche de solutions même lorsqu'un optimum local est rencontré en permettant des déplacements qui n'améliorent pas la solution et en utilisant le principe de mémoire pour éviter les retours en arrière (mouvements cycliques). Contrairement au recuit simulé qui ne génère qu'une seule solution X' aléatoirement dans le voisinage $N(X)$ de la solution courante X, la méthode tabou, dans sa forme la plus simple, examine le voisinage $N(X)$ de la solution courante X. La nouvelle solution X' est la meilleure solution de ce voisinage (dont l'évaluation est parfois moins bonne que X elle-même). Pour éviter de cycler, une liste *taboue* (qui a donné le nom l'algorithme) est tenue à jour et interdit de revenir à des solutions déjà explorées.

c) Recherche à voisinage variable « Variable neighbourhood search »

Propose simplement d'utiliser plusieurs voisinages successifs quand on se trouve bloqué dans un minimum local. On définit un ensemble de k_{max} voisinage, dénotés par N_k=1... k_{max} (N_k Ì N_{k+1}). On choisit une solution de départ X par heuristique. Ensuite, à partir d'une solution initiale X' choisie dans le premier voisinage $N(X)$ de X, on applique une méthode de descente (ou une autre méthode de recherche locale) jusqu'à arriver dans un minimum local (ou que la recherche locale s'arrête). Si la solution trouvée X'' est meilleure que X alors on recentre la recherche en repartant du premier voisinage, sinon on passe au voisinage suivant (qui a priori est plus grand). La recherche s'arrête quand tous les voisinages ne sont plus capables d'améliorer la solution.

d) GRASP

Introduit par Feo et Resende 1995 *Greedy Randomized Adaptative Search procedure* est une métaheuristique multi-départ en deux phases pour la résolution approchée de problèmes NP-difficile de l'optimisation combinatoire. Combine une heuristique gloutonne et une recherche aléatoire. A chaque itération, on construit une solution comme dans une heuristique gloutonne (en se servant d'une liste d'attributs comme liste de priorité). Cette solution est améliorée par l'intermédiaire d'une méthode de descente. En se basant sur la qualité générale de la solution

ainsi obtenue, on met à jour l'ordre de la liste des attributs et le processus est itéré jusqu'à satisfaction d'un critère d'arrêt.

e) recherche locale itérée « Iterated local search »

Dans cette méthode, on génère une solution initiale qui servira de point de départ. Ensuite, on va répéter deux phases : une phase de perturbation aléatoire dans laquelle la solution courante va être perturbée (parfois en tenant compte d'un historique maintenu à jour) et une seconde phase de recherche locale (ou tout simplement de méthode de descente) qui va améliorer cette solution jusqu'à buter sur un optimum local. Dans cette boucle, on est aussi à même d'accepter ou non la nouvelle solution selon que l'on souhaite donner un caractère plus ou moins agressif à la méthode.

f) recherche locale guidée « Guided local search »

Est une variante assez élaborée d'une méthode de descente classique. La méthode de base est simple. Elle consiste à modifier la fonction à optimiser en ajoutant des pénalités. La recherche locale est appliquée alors sur cette fonction modifie. La solution trouvée (qui se trouve être un optimum local) sert à calculer les nouvelles pénalités. Pour cela, on calcule l'utilité de chacun des attributs de la solution et on augmente les pénalités associées aux attributs de valeur maximale. Ces étapes successives sont répétées jusqu'à ce qu'un critère d'arrêt soit valide.

3.2 Métaheuristiques à base de population de solutions

Travaillent sur un ensemble de points de l'espace de recherche en commençant avec une population de solution initiale puis de l'améliorer au fur et à mesure des itérations. L'intérêt de ces méthodes est d'explorer un très vaste espace de recherche et d'utiliser la population comme facteur «de diversité» de plus elle sont très adaptées et très largement utilisées pour l'optimisation multiobjectifs.

3.2.1 Algorithmes Evolutionnaires

Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :
- **une population** constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné, permettant de mémoriser les résultats à chaque étape du processus de recherche.

- **un mécanisme d'évaluation** (fitness) des individus permettant de mesurer la qualité de l'individu,

- **un mécanisme d'évolution** de la population permettant, grâce à des opérateurs prédéfinis (tels que la sélection, la mutation et le croisement), d'éliminer certains individus et d'en créer de nouveaux. Ces méthodes sont applicable dans la plupart des problèmes d'optimisation (multimodaux, non continu, contraints, bruités, multiobjectif, dynamiques, etc.) [Berro, 2001].

On peut distinguer quatre grandes classes d'algorithmes évolutionnaires qui se différencient par leur manière de représenter l'information et par leur façon de faire évoluer la population d'une génération à l'autre :

a) Programmation évolutive (Evolutionary Programming) [Fogel, 1966]

Ce modèle évolutionniste accentue l'utilisation de la mutation et n'utilise pas dans sa version originale la recombinaison des individus par croisement. Développé à l'origine pour l'évolution d'automates à état fini, ce modèle est souvent appliqué à la résolution de problèmes d'optimisation à variables réelles. Dans ce cas, il utilise une mutation qui consiste à ajouter une

perturbation Gaussienne à chaque composante du vecteur à variables réelles constituant l'individu. Cette perturbation est basée sur la performance de l'individu : l'idée consiste à faire subir des mutations importantes aux mauvais individus et inversement des mutation faibles aux bons individus. L'opérateur de sélection est de type probabiliste : il s'agit de la méthode du tournoi basée sur une compétition entre individus choisis aléatoirement.

b) Stratégies d'évolution

Elles ont été développées [Schwefel, 1995] pour résoudre des problèmes d'optimisation industriels à variables réelles dans lesquels il n'existe pas de fonction d'objectif analytique. Ce modèle utilise le principe de mutation sur les réels du modèle de la Programmation Evolutive. Cependant, ce principe a été affiné de sorte que la fonction de perturbation Gaussienne est contrôlée par l'ensemble de la population courante. Si la proportion de mutation réussie est élevée, l'espace de recherche exploré est restreint autour d'un optimum local, il faut donc diversifier la population en augmentant le taux de mutation. Ces approches utilisent un opérateur de sélection de type déterministe : les solutions dont la fitness est mauvaise sont éliminées de la population. En outre, dans le modèle originel, les populations des parents et de leurs descendants sont généralement de taille différente.

c) Algorithmes génétiques (Genetic Algorithms)

En 1809 J.B Lamarck émit l'hypothèse de l'évolution (*les organes d'un animal évolue en fonction de ses besoins)*. En 1859 C. Darwin émit l'idée de la sélection naturelle *(dans tout espèce les meilleurs sont sélectionnés*). Les bases de l'évolution étaient posées. En 1901 De-Vries exposa sa théorie du mutationnisme (*les variations responsables de l'évolution ne se faisaient pas dans le temps mais de façon soudaine et se produisaient dans l'œuf*). Les bases de la génétique étaient posées. Ces concepts vont être utilisées en Informatique : En 1975 Jhon Holland proposa l'Algorithme Génétique. En 1989 Goldberg exposa les fondements mathématiques des AGs.

Le mécanisme des AG [Holland, 1962, Goldberg, 1989, Michalewicz, 1996] consiste à faire évoluer, à partir d'un tirage initial, un ensemble de points de l'espace de recherche vers l'optimum du problème. Avec des opérations extrêmement simples et sans se soucier de la sémantique de ces chromosomes codés en combinant ses solutions entre elles pour en former de nouvelles en essayant d'hériter des bonnes caractéristiques des solutions parents. On obtient une solution qui s'approche de l'optimum en un temps acceptable. C'est en effet cette légèreté de mise en œuvre qui fait la puissance et le charme des AG.

d) Programmation génétique (Genetic Programming)

Est une extension du modèle d'apprentissage des algorithmes génétiques à l'espace des programmes [Koza, 1992]. Les individus formant une population sont donc des programmes candidats à la résolution d'un problème. Ces programmes sont exprimés sous la forme d'arbres sur lesquels les opérateurs génétiques produisent des transformations en vue d'obtenir un programme qui satisfait la résolution du problème choisi.

3.2.2 Colonie de fourmis

Les algorithmes à base de colonies de fourmis ont été introduits par Dorigo. Une des applications principales de la méthode originale était le problème du voyageur de commerce et depuis elle a considérablement évolué. Cette nouvelle métaheuristique imite le comportement de fourmis cherchant de la nourriture. A chaque fois qu'une fourmi se déplace, elle laisse sur la trace de son passage une odeur (la phéromone). Avec plusieurs de ses congénères, elle explore une région en

quête de nourriture. Face à un obstacle, le groupe des fourmis explore les deux côtés de l'obstacle et se retrouvent, puis elles reviennent au nid avec de la nourriture. Les autres fourmis qui veulent obtenir de la nourriture elles aussi vont emprunter le même chemin. Si celui-ci se sépare face à l'obstacle, les fourmis vont alors emprunter préférablement le chemin sur lequel la phéromone sera la plus forte. Mais la phéromone étant une odeur elle s'évapore. Si peu de fourmis empruntent une trace, il est possible que ce chemin ne soit plus valable au bout d'un moment, il en est de même si des fourmis exploratrices empruntent un chemin plus long. Par contre, si le chemin est fortement emprunté, chaque nouvelle fourmi qui passe redépose un peu de phéromone et renforce ainsi la trace, donnant alors à ce chemin une plus grande probabilité d'être emprunté.

3.2.3 Essaim de particulaire [J. Kennedy & R. Eberhart en 1995]

Met en jeu des groupes de particules sous forme de vecteurs se déplaçant dans l'espace de recherche. Chaque particule *p* est caractérisée par deux variables d'état (sa position courante *x(t)* et sa vitesse courante *v(t)*). Cette technique repose sur deux règles :

- Chaque particule se souvient du meilleur point par lequel elle est passée au cours de ses évolutions et tend à y retourner,
- Chaque particule est informée du meilleur point connu au sein de la population et tend à s'y rendre.

3.3 Métaheuristiques avancées « vers les méthodes coopératives »

Les algorithmes à base de population permettent d'explorer un espace de solutions très vaste. Afin d'améliorer les performances d'une recherche globale, de nombreux auteurs proposent d'utiliser une recherche globale pour bien explorer l'espace de recherche conjointement avec une recherche locale pour mieux exploiter les zones prometteuses. Exemple en ajoutant des techniques de recherche locale aux algorithmes génétiques, on obtient les algorithmes **mémétiques**.

a) Algorithme mémétique

Moscato introduit en 87 pour la première fois les algorithmes mémétiques. L'idée principale de cette technique est de rendre plus agressif un algorithme génétique par l'ajout d'une recherche locale en plus de la mutation. On rencontre aussi le nom *genetic local search*. Une des observations générales provenant de l'implémentation d'un algorithme génétique basique est souvent la faible vitesse de convergence de l'algorithme. L'idée de Moscato est donc d'ajouter une recherche locale qui peut être une méthode de descente ou une recherche locale plus évoluée (recuit simule ou recherche tabou par exemple). Cette recherche locale sera appliquée à tout nouvel individu obtenu au cours de la recherche.

b) Scatter search Recherche par dispersion

Une population de solutions (assez importante au départ) est générée (en essayant de proposer des solutions diverses les unes des autres) et chaque individu est amélioré par l'application d'une recherche locale. De cette population on extrait un *ensemble de référence* (*Reference set*, *R*) contenant les meilleures solutions de la population initiale. Ensuite ces solutions sont combinées entre elles (et avec les nouvelles solutions générées) puis améliorées jusqu'à ce qu'il n'y ait plus de nouvelles solutions générées par combinaison. Ensuite une moitié de la population est régénérée (remplacée par des solutions diverses) et le processus recommence jusqu'à satisfaction d'un critère d'arrêt. Un des points importants, c'est la mesure de la diversité des solutions. Cette mesure doit être prise dans l'espace des solutions (et non dans l'espace des objectifs) et elle doit

refléter la différence entre deux solutions. La particularité de cette méthode est l'acharnement à épuiser les ressources par combinaison. Ceci se traduit par une sorte d'exploration systématique de tout un voisinage possible et bien sûr par un temps d'exécution parfois prohibitif.

c) MA/PM

En partant des observations faites sur la plupart des métaheuristiques précédentes, y compris des plus sophistiquées, plusieurs caractéristiques importantes sont apparues à Sörensen. En premier lieu, une recherche locale est indispensable. De même, il est important de mesurer (comme dans le scatter search) la diversité des solutions. Par ailleurs, l'exploration systématique n'est pas toujours nécessaire. De ces constatations, Sevaux et Sörensen ont proposé cette nouvelle métaheuristique appelée MA/PM. Le fonctionnement est assez simple et est basé sur un algorithme génétique. Nous supposons que nous savons comparer deux individus entre eux et mesurer leur dissemblance. Nous pouvons donc mesurer la similarité entre un individu et la population existante. Au départ, on génère une population initiale de petite taille et on choisit un paramètre Δ fixant le niveau de dissemblance des solutions entre elles. Ensuite, on procède comme dans un algorithme génétique, on choisit deux individus que l'on croise pour obtenir deux enfants. Pour chacun on applique une recherche locale de façon à obtenir des optima locaux. S'ils ne répondent pas au critère de diversité, on applique un opérateur de mutation sur ces individus jusqu'à satisfaction de ce critère. Ensuite sous condition, on les insère dans la population à la place d'un autre individu. A chaque itération le paramètre Δ gérant la diversité est mis à jour. Ce qui fait sans doute que cette méthode donne de très bons résultats, c'est qu'elle combine les avantages du scatter search et d'un algorithme mémétique. D'un côté elle applique une recherche locale à toutes les nouvelles solutions et de l'autre, elle maintient une population de petite taille et diversifiée. A la différence du scatter search, la diversité est contrôlée précisément. L'autre avantage, c'est l'évolution du paramètre de diversité Δ (comme dans les schémas de refroidissement de température du recuit simulé) qui permet à tout moment d'augmenter ou de réduire la diversité des individus dans la population.

4 Les méthodes hybrides

L'idée de faire coopérer différents types de méthodes n'est pas nouvelle. Très vite, il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés et on a cherché à profiter des avantages des différentes méthodes.

4.1 Coopération méta/méta

Les coopérations étaient à l'origine essentiellement réalisées entre différentes métaheuristiques. A peu prés tous les types d'approches ont été proposés pour ce type de coopération, fait que les métaheuristiques hybrides sont devenues maintenant assez classiques dans le domaine de l'optimisation [Basseur, 2005].

4.2 Coopération méta/exacte

Nous avons vu que les méthodes exactes permettaient de résoudre des petits problèmes tandis que les (méta)heuristiques sont capables d'appréhender de grands problèmes sans pouvoir donner la solution optimale ou prouver que la solution fournie est optimale. Cependant les méthodes exactes peuvent néanmoins être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution

globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet [Dhaenens 2005]. Cette constatation constitue le point de départ d'une approche hybride, assez originale dans le contexte des problèmes d'optimisation combinatoire visant à combiner résolutions exactes et métaheuristiques permettant de conserver aux mieux les avantages de chacune des approches. Nous nous intéressons à **la coopération méta/exacte**, afin d'obtenir toujours de meilleurs résultats.

5 Petit état de l'art sur les méthodes de résolution multiobjectif

5.1 Méthodes exactes pour l'optimisation multiobjectif

Dans la littérature, plusieurs méthodes exactes multiobjectifs basées sur le branch and bound [Sen, 1988], sur l'algorithme A* [Stewart, 1991] et la programmation dynamique [Carraway, 1990], la méthode à deux phases [Ulungu, 1995] ont été proposées pour résoudre de petits problèmes à deux objectifs. Pour les problèmes à plus de deux critères ou de grandes tailles, il n'existe pas de procédures exactes efficaces, étant données les difficultés simultanées de la complexité NP-difficile, et le cadre multiobjectif des problèmes. Ainsi, on assiste ces dernières années à un accroissement de intérêt porté sur l'utilisation de métaheuristiques pour la résolution de Problèmes Multiobjectifs, et spécialement de problèmes réels.

5.2 Métaheuristiques pour l'optimisation multiobjectif

Un livre publié par Coello Coello et Lamont montre le spectre des applications multiobjectif abordées par les algorithmes évolutionnaires [Coello, 2004]. Ces applications vont du design d'appareils électromagnétiques, au design de circuits logiques, à l'optimisation de problèmes de tournées en passant par l'analyse de promoteurs dans le domaine de la bioinformatique. Chaque application apportant de nouveaux challenges, de nouveaux opérateurs et de nouvelles stratégies où combinaisons de méthodes sont proposés. L'approche Pareto a été introduite initialement dans les algorithmes génétiques par Goldberg [Goldberg, 1989]. L'utilisation d'une sélection basée sur la notion de dominance de Pareto va faire converger la population vers un ensemble compromis de solutions (le front Pareto). D'une manière générale, la plupart des approches Pareto trouvées dans la littérature sont incluses dans des algorithmes évolutionnaires, et la plupart de ces algorithmes évolutionnaires sont des algorithmes génétiques.

– NSGA: Nondominated Sorting Genetic Algorithm by Srinivas and Deb, 94.
– NPGA: Niched Pareto Genetic Algorithm by Horn, Nafpliotis and Goldberg, 94.
– MOGA: Multi-ObjectiveGenetic Algorithm by Fonseca & Fleming 93. Murata & Ishibuchi 95
– MGK: Morita-G.-Katoh Algorithm by Morita & Katoh, 98, 2001.
– SPEA: Strength Pareto Evolutionary Algorithm by Zitzler & Thiele, 98.
– PAES: Pareto Archived Evolution Strategy by Knowles & Corne, 99.

Ainsi, les algorithmes évolutionnaires dédiés à l'optimisation combinatoire multiobjectif ont bien évolué ces dernières années. Cependant ces méthodes ne restent que des approches heuristiques et si sur des applications réelles elles semblent être performantes, l'objectif est toujours de chercher à améliorer les résultats. Dans cet objectif, une approche prometteuse concerne la coopération de différentes méthodes et en particulier la coopération entre méthodes exactes et méthodes heuristiques [Dhaenens, 2005].

5.3 Coopération de méthodes pour l'optimisation multiobjectif

En optimisation multiobjectif, la coopération est plus récente et concerne essentiellement la coopération entre méthodes heuristiques.

- MO-GLS: Multi-Objective Genetic Local Search by Jaszkiewicz, 2001.
- MOGTS: Multi-Objective Genetic Tabu Search by Barichard & Hao, 2002.
- Ishibushi et Yoshida proposent pour le flowshop de faire coopérer des algorithmes évolutionnaires multiobjectifs (SPEA, NSGA II) en utilisant de la recherche locale en tant qu'opérateur de mutation.

En multiobjectif peu de travaux méta/exacte ont été réalisés. On peut citer les travaux de M.Basseur dans sa thèse de doctorat [Basseur, 2005] qui présente trois méthodes réalisées entre un AG Adaptatif et la méthode à deux phases. Et Nicolas Jozefowiez [Jozefowiez, 2004] dans sa thèse propose une méthode coopérative entre un algorithme génétique et un algorithme de séparation et coupes (Branch and Cut).

Chapitre 2

Deuxième partie

Dans cette partie du chapitre, nous présentons quelques méthodes de résolution Pareto fondées sur des métaheuristiques.

Sommaire

6 Les principales métaheuristiques pour l'optimisation multiobjectif

Les méthodes à base de populations travaillant avec un ensemble de solutions potentielles, tels que les algorithmes évolutionnaires, sont bien adaptées à ce type de problème [Coello, 2002]. C.A. Coello Coello, D.A. Van Valdhuisen et G. B. Lamont proposent une classification de ces algorithmes évolutionnaires selon deux générations de méthodes. Ce qui différencie la deuxième génération est la présence d'une population secondaire (archive) et de méthodes de recherche avancées.

6.1 Les techniques Non-élitiste

Les approches que nous allons voir ne conservent pas les individus Pareto optimaux trouvés au cours du temps. Elles maintiennent difficilement la diversité sur la frontière Pareto. La convergence des solutions vers la frontière de Pareto est lente. [Berro, 2001]

6.1.1 Multiple Objective Genetic Algorithm (MOGA)

Fonesca et Fleming [Fonseca, 1995] ont proposé une méthode dans laquelle le rang d'un individu est proportionnel au nombre d'individus le dominant : $r_i = 1 + Dom(i)$. Tous les individus non dominés sont de rang 1. La fitness de chaque individu est calculé par application d'une fonction de *scaling* sur la valeur de son rang. Cette fonction est en général linéaire.
L'utilisation de la sélection par rang a tendance à répartir la population autour d'un même optimum. Or cela n'est pas satisfaisant pour un décideur car cette méthode ne lui proposera qu'une seule solution. Pour éviter cette dérive, les auteurs utilisent une fonction de *sharing*, ils espèrent ainsi répartir la population sur l'ensemble de la frontière de Pareto. Le *sharing* utilisé dans cette méthode agit sur l'espace des objectifs. Cela suppose que deux actions qui ont le même résultat dans l'espace des objectifs ne pourront pas être présentés dans la population. Cette méthode obtient des solutions de bonne qualité et son implémentation est facile. Mais les performances sont dépendantes de la valeur du paramètre s_{share} utilisé dans le sharing. Dans son article Fonesca explique comment choisir au plus juste la valeur s_{share}.

6.1.2 Non dominate Sorting Genetic Algorithm (NSGA)

Dans la méthode proposée par [Srinivas, 1993], le calcul de la de la fitness s'effectue en séparant la population en plusieurs groupes en fonction du degré de domination au sens de Pareto de chaque individu.

Algorithme de la fonction de notation

- Dans la population entière, on recherche les individus non dominés. Ces derniers constituent la première frontière de Pareto.
- On leur attribue une fitness factice. Cette valeur est supposée donner une chance égale de reproduction à tous ces individus. Mais pour maintenir la diversité dans la population, il est nécessaire d'appliquer une fonction de sharing sur cette valeur.
- Ce premier groupe d'individus est supprimé de la population
- On recommence cette procédure pour déterminer la seconde frontière de Pareto. La valeur factice de fitness attribuée à ce second groupe est inférieure à la plus petite fitness après application de la fonction de sharing sur le premier groupe. Ce mécanisme est répété jusqu'à ce qu'on ait traité tous les individus de la population.
- L'algorithme se déroule ensuite comme un algorithme génétique classique. Srinivas utilise une sélection basée sur le reste stochastique. Mais sa méthode peut être utilisée avec d'autres heuristiques de sélections (tournoi, roulette, etc.)

Cette méthode paraît moins efficace en temps de calcul que la méthode MOGA car le temps de calcul de la notation (trie de la population et sharing) est important. Mais cette technique semble plus appropriée à maintenir une grande diversité de la population et à répartir plus efficacement les solutions sur la frontière de Pareto.
Trois critiques ont été soulevées pour cette méthode [Berro, 2001] :

- Sa complexité de calcul de $O(k,N^3)$ avec k le nombre d'objectifs et N la taille de la population, essentiellement due au processus de trie de la population et d'application de l'heuristique de partage
- Son approche non élitiste.
- La nécessité de spécifier un paramètre de sharing.

6.1.3 Niched Pareto Genetic Algorithm (NPGA)

Cette méthode proposée par Horn et Napfliotis [Horn, 1993] utilise un tournoi basé sur la notion de dominance de Pareto. Elle compare deux individus pris au hasard avec une sous-population de taille t_{dom} également choisie au hasard. Si un seul de ces deux individus domine le sous-groupe, il est alors positionné dans la population suivante. Dans les autres cas une fonction de sharing est appliquée pour sélectionner l'individu. Le paramètre t_{dom} permet d'exercer une pression variable sur la population et ainsi d'augmenter ou de diminuer la convergence de l'algorithme. A travers leurs expérimentations Horn et Napfliotis établissent le constat suivant :

- Si $t_{dom} \approx 1\%$ de N, il y a trop de solutions dominées.
- Si $t_{dom} \approx 10\%$ de N, une bonne distribution des individus est obtenue.
- Si $t_{dom} >> 20\%$ de N, il y a une convergence prématurée, car la pression est trop importante lors de la sélection.

Outre les paramètres de sharing l'utilisation de cette méthode ajoute un paramètre supplémentaire à fixer par l'utilisateur, t_{dom}. Ce dernier permet de régler aisément la pression sur la population. Les solutions trouvées sont de bonne qualité et cette approche est plus rapide que les approches précédentes car le sharing n'est appliqué que sur une portion de la population.

6.2 Les techniques élitiste

Dans le domaine de l'optimisation multiobjectif, la sélection élitiste consiste à maintenir une seconde population appelée archive, contenant les solutions non dominées trouvées au cours des différentes générations de l'algorithme évolutionniste. Les individus de cette population participe avec une certaine probabilité à l'étape de sélection et donc à la reproduction de nouveaux individus. En outre des techniques de regroupements ou *clustering* sont employées pour limiter la taille de cette archive. Cette technique est utilisée pour résoudre les difficultés des méthodes non-élitistes.

6.2.1 Strength Pareto Evolutionary Algorithm (SPEA)

En 1998 Zitzler et Thiele proposent la méthode élitiste SPEA [Zitzler, 1998] basée sur le concept Pareto. Pour réaliser cet élitisme, SPEA maintient une archive externe contenant le meilleur front de compromis rencontré durant la recherche. Toutes les solutions de l'archive participe à la sélection. Une méthode de clustering est utilisée pour réduire l'ensemble de Pareto sans supprimer ses caractéristiques. Une nouvelle méthode de niche, basée sur Pareto, est utilisée afin de préserver la diversité. L'avantage essentiel est qu'elle n'exige pas de réglage de paramètre de sharing.

Fonctionnement général

La première étape consiste à créer une population initiale (constituée par exemple d'individus générés aléatoirement). L'archive externe, au départ initialisée à l'ensemble vide, est mise à jour régulièrement en fonction des individus non dominés de la population. À chaque itération de l'algorithme, on retrouve les étapes classiques sélection, croisement et mutation. Puis les nouveaux individus non dominés découverts viennent s'ajouter à l'archive, et les individus de l'archive dominés par le nouvel arrivant sont supprimés. Si l'archive vient à excéder une certaine taille (fixée au départ), alors une phase de clustering est appliquée dans le but de garder les meilleurs représentants.

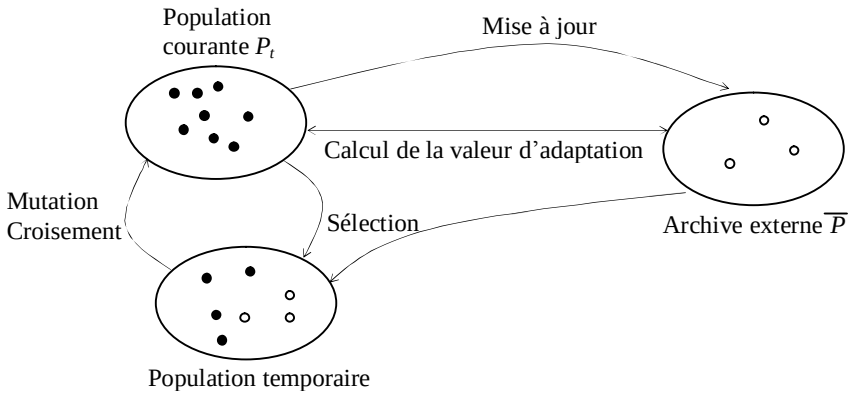


Figure 2.2 Fonctionnement général de l'algorithme SPEA.

Algorithme SPEA.

- Initialiser la population P_0 et créer l'archive externe vide $\overline{P} = \varnothing$;
- Mise à jour de P à partir des individus non dominés de P_0
- Tant que critère d'arrêt non rencontré faire
    - Calcul de la valeur d'adaptation pour tous les individus de P + $\overline{P}$
    - Sélection dans P_t + $\overline{P}$ en fonction de la valeur d'adaptation
    - Croisement
    - Mutation
    - Mise à jour de $\overline{P}$ à partir des individus non dominés de P

FinTantque

Calcul de la valeur d'adaptation

Le calcul de la valeur d'adaptation de SPEA permet de diminuer les chances de sélection des individus ayant une moins bonne évaluation, ou des individus issus de niches déjà importantes. A la différence des techniques classiques de *sharing* qui sont basées sur la notion de distance euclidienne dans l'espace des objectifs, SPEA utilise la notion de dominance pour détecter les niches d'individus. De plus, de par la nature élitiste de l'algorithme, les individus de l'archive externe participent eux aussi à la sélection, le calcul d'adaptation doit donc les prendre en compte.

Ce calcul s'effectue en deux étapes. La première étape consiste à affecter une valeur, appelée valeur de dureté (S), aux individus de l'archive $\overline{P}$ externe. Cette valeur est déduite à partir du nombre d'individus qu'un élément i ∈ $\overline{P}$ domine faiblement dans la population Courante Pt :

$$S(i) = \frac{\left|\left\{j / j \in P_t \wedge f(i) \leq f(j)\right\}\right|}{\left|P_t\right| + 1}$$

La valeur d'adaptation d'un individu $i \in \overline{P}$ est égale à sa valeur de dureté : *F(i) = S(i)*. Pour un individu j de la population courante Pt, la valeur d'adaptation est calculée en réalisant la somme des valeurs de dureté des individus de P dominant j :

$$F(j) = 1 + \sum_{i \in \overline{P} \wedge f(i) \leq f(j)} S(i)$$

Le chiffre 1 est ajouté au total de la somme pour éviter que des individus de $\overline{P}$ aient une valeur d'adaptation plus grande que certains individus de P_t. Il est à noter que, dans ce calcul, une plus petite valeur d'adaptation conduit à une plus grande chance de sélection de l'individu. Ainsi, plus un individu est dominé par les individus de $\overline{P}$ et plus sa valeur d'adaptation décroît, diminuant donc ses chances d'être sélectionné. Un exemple de calcul de valeurs d'adaptation est illustré à la *figure* 2.3. En observant cette figure, il est clair que l'individu évalué à 19/6 (qui est le résultat de la somme : 1+3/6+2/6+3/6+3/6+2/6) à une probabilité moindre d'être sélectionné que les individus évalués à 14/6.

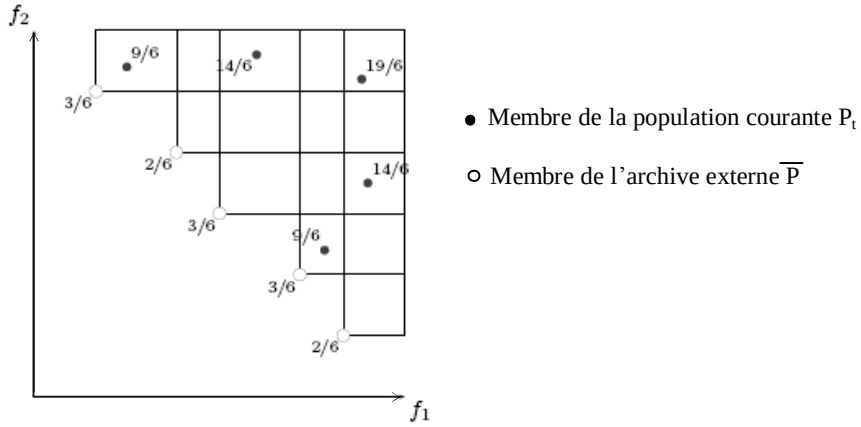


Figure 2.3 calcul des valeurs d'adaptation en dimension 2.

Réduction par clustering

Lorsque le nombre d'individus de l'archive externe est grand, les performances de l'algorithme peuvent se dégrader significativement. En effet, le nombre d'individus influe sur le calcul de la valeur d'adaptation, celui-ci devient moins fiable, et peut tromper la recherche en la focalisant, par exemple, sur des zones déjà explorées.

Pour pallier à ce défaut, une solution consiste à utiliser des techniques de *clustering*. Ces techniques ont été étudiées intensivement dans le contexte des analyses de cluster [Morse, 1980] et ont été appliquées avec succès pour déterminer des partitions d'une collection relativement hétérogène d'éléments. La technique de *clustering* utilisée par SPEA est assez intuitive. Au début de la procédure, chaque individu constitue son propre groupe, puis on fusionne deux à deux les groupes les plus proches en terme de distance. Cette étape est itérée jusqu'à l'obtention du nombre désiré de groupes.

Une fois les groupes identifiés, il ne reste plus qu'à choisir un représentant par groupe. Ce représentant peut être déterminé de plusieurs façons, par exemple en prenant le barycentre du groupe. C'est ce représentant qui sera gardé, les autres éléments étant tout simplement supprimés. Les étapes de la méthode de clustering sont illustrées à la *figure* 2.4.

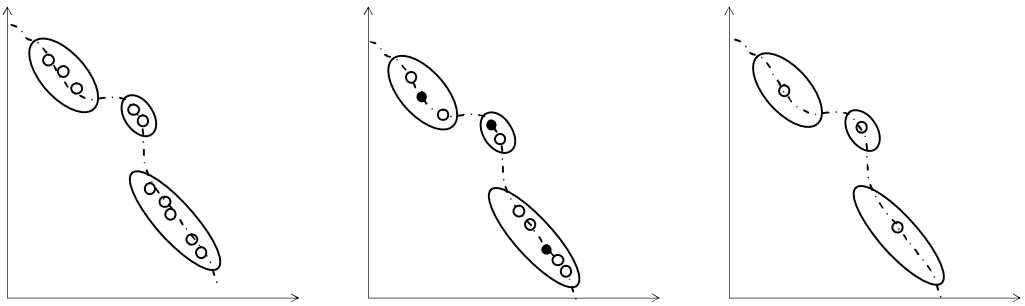


Figure 2.4 Illustration du clustering en dimension 2.

6.2.2 Pareto Archived Evolution Strategy (PAES)

Cette méthode a été développée par Corne et Knowless [Knowless, 1999]. Elle n'est pas basée sur une population mais, utilise une population annexe de taille déterminée permettant de stocker les solutions temporairement Pareto optimales.

Algorithme PAES.

- Génération d'une solution candidate.
- Fonction d'acceptation de la solution candidate.
- Archivage des solutions non dominées.

La méthode de génération d'un nouveau candidat

- Génération aléatoire d'une solution ***c*** et ajout de ***c*** à l'archive
- Production d'une solution ***m*** par mutation de ***c*** et évaluation de ***m***
- Si ***c*** domine ***m***
  Alors on écarte ***m***
  Sinon si ***m*** domine ***c***
  Alors on remplace ***c*** par ***m*** ajout de ***m*** dans l'archive
  Sinon si ***m*** est dominé par un membre de l'archive
  Alors on écarte ***m***
  Sinon **fonction test**(***c***, ***m***, archive) % détermine la nouvelle solution courante
- On recommence à la 2ème ligne.

Fonction test(c, m, archive)

- Si l'archive n'est pas pleine
  Alors ajout de ***m*** dans l'archive
  Sinon si ***m*** est dans une région moins encombrée qu'une solution $\boldsymbol{x} \in$ à l'archive
  Alors ajout de ***m*** et suppression d'un membre de la zone la plus encombrée de l'archive.
- Si ***m*** est dans une région moins encombrée que ***c***
  Alors ***m*** est acceptée comme solution courante
  Sinon on conserve ***c*** comme solution courante

Pour mesurer l'encombrement d'une zone, cette méthode utilise une technique de l'étalement (*crowding*) basée sur un découpage en hypercubes de l'espace des objectifs.

Cette méthode est relativement simple à mettre en œuvre. De plus, n'étant pas basée sur un algorithme génétique, elle évite à l'utilisateur le réglage de tous les paramètres de celui-ci. Son efficacité va dépendre du choix du nouveau paramètre de discrétisation de l'espace des objectifs. La technique de crowding utilisée offre deux avantages par rapport aux méthode de sharing classiques : le temps de calcul est moins important et la découpage étant adaptatif, cela ne nécessite pas de réglage de paramètre.

6.2.3 Pareto Envelope based Selection Algorithm (PESA)

Propose également par Corne et Knowless [Knowless, 2000]. Elle reprend approximativement le principe de crowding de PAES et définit un paramètre appelé *squeeze_factor* qui représente la mesure d'encombrement d'une zone de l'espace. Alors que PAES est basée sur une stratégie d'évolution, PESA est une méthode basée sur les algorithmes génétiques.

Algorithme PESA.

- Génération aléatoire et évaluation de la population P_I, et initialisation de P_E=∅
- Transfert de tous les individus non dominés de P_I dans P_E.
- Si le critère d'arrêt est atteint
  Alors on retourne P_E comme ensemble de solution
  Sinon on suprime tous les individus de P_I et on recrée P_I de la façon suivante :
    On sélectionne 2 parents dans P_E avec la probabilité p_c,
    On produit un enfant par croisement puis on le mute.
    Avec la probabilité (1-p_c), on sélectionne un parent et on le mute pour produire un autre enfant.
- On recommence à la 2ème ligne.

Une solution courante de P_I peut entrer dans l'archive P_E si elle est non dominée dans P_I et si elle est non dominée dans P_E. Une fois la solution insérée dans l'archive, on supprime tous les membre de l'archive qu'elle domine. Si l'ajout crée un dépassement de capacité de P_E alors le membre de l'archive ayant le paramètre de *squeeze_factor* le plus élevé est supprimé. Ce paramètre est égal au nombre d'individus qui appartiennent au même hypercube.

6.2.4 Non dominate Sorting Genetic Algorithm II (NSGAII)

Dans cette deuxième version de NSGA [Deb, 2000], l'auteur tente de résoudre toutes les critiques faites sur NSGA : complexité, non élitiste et utilisation de sharing. NSGA-II intègre un opérateur de sélection, basé sur un calcul de la distance de *crowding*, très différent de celui de NSGA. Comparativement à NSGA, NSGA-II obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de K.Deb, ce qui fait de cet algorithme un des plus utilisés aujourd'hui [Barichard, 2003]. NSGA-II est un algorithme élitiste n'utilisant pas d'archive externe pour stocker l'élite. Pour gérer l'élitisme, NSGA-II assure qu'à chaque nouvelle génération, les meilleurs individus rencontrés soient conservés. Pour comprendre le fonctionnement de l'algorithme, plaçons nous à la génération *t*. Comme le montre la *figure* 2.5, deux populations (P_t et Q_t de taille N) coexistent. La population P_t contient les meilleurs individus rencontrés jusqu'à la génération *t*, et la population Q_t est formée d'individus autres, issus des phases précédentes de l'algorithme.

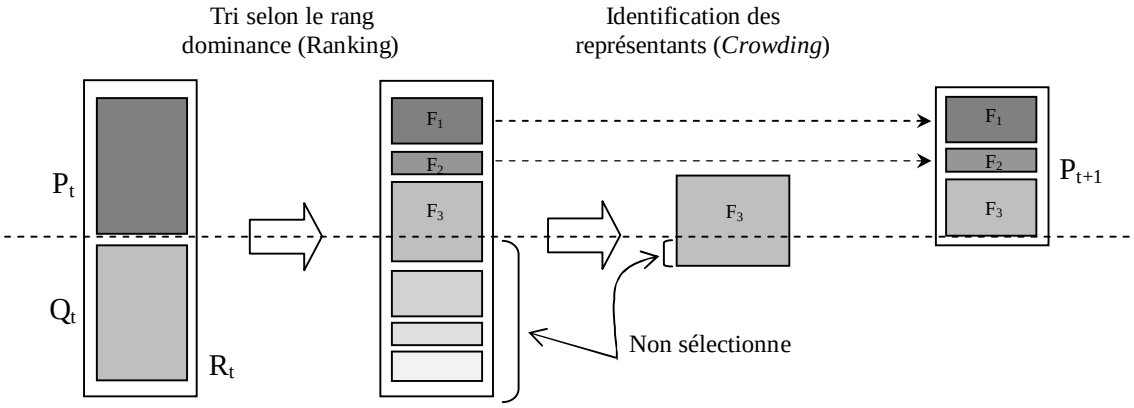


Figure. 2.5 Schéma de fonctionnement de NSGA-II

Algorithme NSGA-II

- Initialiser les populations P_0 et Q_0 de taille N
- Tant que critère d'arrêt non rencontré faire
  - Création de $R_t = P_t \cup Q_t$
  - Calcul des différents fronts Fi de la population Rt par un algorithme de \ranking"
  - Mettre $P_{t+1} = \emptyset$ et i = 0,
  - Tant que $|P_{t+1}| + |F_i| < N$ faire
    $P_{t+1} = P_t \cup F_i$
    i = i + 1
    FinTantQue
  - Inclure dans P_{t+1} les (N - $|P_{t+1}|$) individus de F_i les mieux répartis au sens de la distance de crowding
  - Sélection dans P_{t+1} et création de Q_{t+1} par application des opérateurs de croisement et mutation
- FinTantQue

Le calcul de valeur d'adaptation pour NSGA-II ne sert pas uniquement pour la sélection des opérateurs de croisement et de mutation, mais intervient aussi dans la sélection des individus à inclure dans P_{t+1} (la population contenant les élites). C'est donc une phase importante pour laquelle les auteurs de NSGA-II ont développé une méthode particulière : la distance de crowding.

Calcul de la distance de crowding

La distance de crowding d_i d'un point particulier i se calcule en fonction du périmétre de l'hypercube ayant comme sommets les points les plus proches de i sur chaque objectif. Sur la *figure* 2.6 est représenté l'hypercube en deux dimensions associé au point i. Un algorithme de calcul de la distance de crowding est détaillé dans [Deb, 2001]. Cet algorithme est de complexité $O(MN \log(N))$, où M est le nombre d'objectifs du problème et N le nombre d'individus à traiter. Une fois tous les d_i calculés, il ne reste plus qu'à les trier par ordre décroissant et à sélectionner les individus possédant la plus grande valeur de crowding.

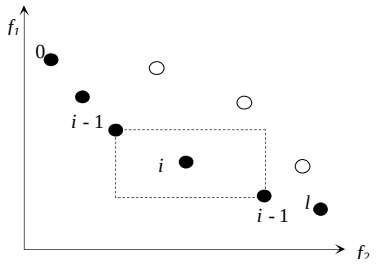


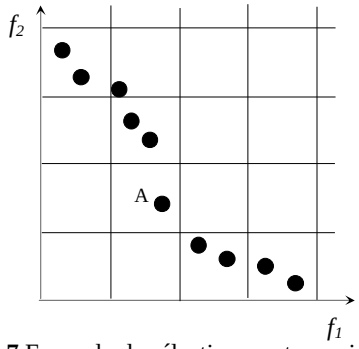
Figure 2.6 Calcul d'un représentant (*crowding*)

Cette nouvelle version de NSGA a permis de réduire la complexité de l'algorithme à $O(k,N^2)$, de créer une méthode plus élitiste et de supprimer les paramètres de sharing.

6.2.5 PESA II: Region-Based Selection

PESA II est une nouvelle technique de sélection basée sur l'utilisation d'hypercubes dans l'espace des objectifs [corne, 2001]. Qu lieu d'effectuer une sélection en fonction de la fitness des individus comme dans PESA, cette méthode effectue une sélection par rapport aux hypercubes occupés par au moins un individu. Après avoir sélectionné l'ypercube, on choisit aléatoirement l'individu dans l'hypercube. Cette méthode se montre plus efficace à répartir les solutions sur la frontière Pareto. Cela est dû à sa capacité de choisir avec une plus grande probabilité que le tournoi classique, des individus situés dans des zones désertiques.
Par exemple dans la figure ci-dessous, les 10 points sont répartis dans 6 cubes. Si l'on considère un tournoi binaire alors la probabilité de sélectionner la solution A est :

- Dans PESA : $1 - (9/10)^2 = 0,19$
- Dans PESA II : $1- (5/6)^2 = 0,31$



Si le découpage avait été de 4 cubes au lieu de 16 alors la probabilité de sélectionner la solution A serait passée de 0,31 à 0,89. Cela montre la très forte influence de la discrétisation de l'espace sur cette méthode

Figure 2.7 Exemple de sélection par tournoi hypercube

PESA II a permis de faire évoluer positivement la sélection de manière à privilégier les zones de l'espace les moins encombrées. Mais, même si cette technique de crowdin basée sur un découpage de l'espace est très supérieure en temps de calcul au sharing, elle possède ses propres difficultés.

7 Conclusion

Dans ce chapitre nous avons présenté des méthodes d'optimisation mono et multiobjectifs nécessaires à la compréhension de la suite du travail rapporté dans ce mémoire. Le choix de la méthode de résolution à mettre en œuvre dépendra souvent de la complexité du problème. Si le problème est de petite taille, alors un algorithme exact permettant de trouver la solution optimale peut être utilisé. Malheureusement, ces algorithmes par nature énumératifs, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles. Dans ce cas, il est nécessaire de faire appel à des (méta)heuristiques permettant de trouver de bonnes solutions approchées. Il est apparu que toutes les méthodes n'avaient pas les mêmes propriétés et on a cherché à profiter des avantages des différentes méthodes. L'activité s'est focalisée donc, sur le développement de nouvelles approches algorithmiques hybrides. L'intérêt de ces approches coopératives est de permettre à différentes méthodes d'optimisation d'allier leurs

atouts dans le but d'améliorer les performances globales obtenues par chacune d'elles, afin d'obtenir de bon résultats. Les coopérations étaient à l'origine essentiellement réalisées entre différentes métaheuristiques. Néanmoins, Les méthodes exactes peuvent être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet, Cette constatation constitue le point de départ d'une approche hybride, assez originale dans le contexte des problèmes d'optimisation combinatoire visant à combiner résolutions exactes et métaheuristiques permettant de conserver au mieux les avantages de chacune des deux méthodes.

Pour répondre à l'objectif de concevoir un algorithme efficace, en temps de calculs, en qualité de solutions produites (l'intensification et la diversification) et permettant de traiter des problèmes de plus grande taille. Nous avons adopté la méthode hybride méta/exacte, réalisée entre un algorithme de type B&B et la métaheuristique MA|PM que nous avons adapté au cas multiobjectif.

Chapitre 3

Coopération entre méthodes exactes et métaheuristiques

Dans ce chapitre, nous présentons des classifications de la méthode coopérative méta/exacte avec des exemples trouvés dans la littérature

Sommaire

1 Introduction

Durant les dernières années, l'activité s'est focalisée sur le développement d'approches algorithmiques hybrides pour la résolution aussi bien pour des problèmes NP-difficiles académiques de taille de plus en plus importante, que pour des problèmes issus de systèmes réels de plus en plus complexes et multiobjectifs. L'intérêt de ces approches coopératives est de permettre à différentes méthodes d'optimisation d'allier leurs atouts dans le but d'améliorer les performances globales obtenues par chacune d'elles afin d'obtenir de bon résultats par rapport aux méthodes qui les composent. Actuellement les meilleurs résultats obtenus sont issus de ce type d'approche, en particulier sur les problèmes réels [Basseur, 2005].

Pour les problèmes à plus de deux objectifs ou de grande taille, il n'existe pas de méthodes exactes efficaces, étant donné les difficultés simultanées de la complexité NP-difficile, et le cadre multiobjectif des problèmes. Afin de résoudre des problèmes de grande taille et/ou des problèmes avec plus de deux objectifs, les méthodes exactes peuvent néanmoins être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet [Dhaenens 2005]. Ces travaux sont généralement efficaces, car les deux méthodes combinées ont alors des particularités bien différentes, et associent leurs avantages. Par exemple, une fois des régions de bonne qualité localisées rapidement par un algorithme à base de population il peut être intéressant de poursuivre la recherche en appliquant une méthode exacte sur les solutions de voisinage. Des heuristiques peuvent être utilisées afin d'accélérer l'énumération des solutions en trouvant de bonnes bornes, en offrant des solutions initiales ou en définissant des plans de coupes prometteurs, aux méthodes exactes.
Dans les sections suivantes, nous allons présenter l'état de l'art de combiner des algorithmes exacts et des métaheuristiques pour résoudre des problèmes d'optimisation combinatoire avec des exemples de la littérature déjà réalisés.

2 Coopération méta/exacte « état de l'art »

Un état de l'art de ces coopérations a été proposé par Stützle et Dumitrescu [Dumitrescu, 2003]. Ils distinguent cinq types d'approches :
– Utilisation d'un algorithme exact pour explorer des larges voisinages dans une recherche locale.
– Utilisation des solutions de bonne qualité afin de réduire l'espace de recherche de la méthode exacte.
– Exploitation des bornes de la méthode exacte pour une heuristique constructive.
– Utilisation des informations fournies par les relaxations des problèmes linéaires pour orienter un algorithme de recherche local ou constructif.
– Utilisation d'une méthode exacte pour une fonction spécifique de la métaheuristique.

Une approche plus générale de la classification de type taxonomie a été proposée par M.Basseur [Basseur, 2005] complétée à partir de la structure générale de la taxonomie des métaheuristiques hybrides réalisée antérieurement par EG.Talbi. Cette approche est composée des modèles de classification **hiérarchiques** et des modèles de classification **à plat**. Une autre classification

proposée par J.Puchinger et G.R.Raidl [Puchinger, 2005] se résume en deux catégories principales, **collaborative** et **intégrative**.

2.1 Classification hiérarchique

Dans cette classification les propriétés discriminantes sont le niveau de coopération et le mode de coopération.

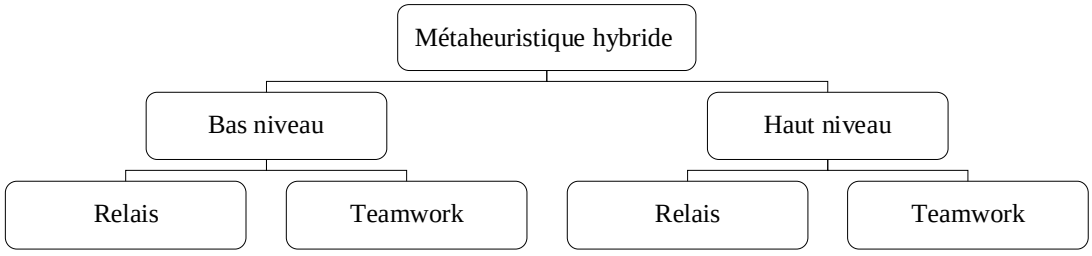


Figure 3.1 Structure de la taxonomie hiérarchisée

a) **Niveau de coopération** : On distingue les hybridations de ***bas niveau*** et des hybridations de ***haut niveau***.

- Dans l'hybridation de bas niveau une fonction interne d'une méthode est remplacée par une autre méthode.
- Dans l'hybridation de haut niveau la structure interne des méthodes hybridées n'est pas modifiée

b) **Mode de coopération** : On distingue les hybridations en mode ***relais*** et des hybridations en mode ***coévolution (teamwork)***.

- En mode relais, les méthodes coopératives opèrent les une après les autres dans un ordre prédéterminé. La première, chaque méthode impliquée dans la coopération reçoit en entrée le résultat produit par la précédente.
- En mode teamwork plusieurs agents coopèrent en parallèle. Chaque méthode hybridée (agent) conduit une recherche dans un espace de solution donné.

Selon cette classification, nous obtenons quatre classes des algorithmes hybrides :

1) **La classe LRH**, coopération de bas niveau en mode relais (Low-level Relay Hybrid), regroupe les coopérations constituées d'une méthode d'optimisation à solution unique dans laquelle est insérée une autre méthode d'optimisation [Basseur, 2005]. Ce type de coopération est plus facilement envisageable en ayant une méthode heuristique au service d'une méthode exacte. Exemple dans [Augerat, 98] un algorithme de B&C est proposé pour résoudre un problème de tournées de véhicules avec contraintes de capacité. Les auteurs partent de la remarque que les inégalités du programme linéaire traitant des contraintes de capacité sont celles qui sont à l'origine des meilleurs plans de coupes, ce qui permet les meilleures améliorations de la borne lorsque ces contraintes sont ajoutées à la formulation du programme linéaire relaxé. Trois approches heuristiques, une constructive, une gloutonne et une tabou, pour extraire un ensemble pertinent de contraintes de capacité violées du problème relaxé.

2) La classe LTH, coopération de bas niveau en mode teamwoke (Low-level teamwork Hybrid), regroupe les coopérations constituées d'une méthode d'optimisation à population de solutions pour laquelle un opérateur agissant sur les solutions de manière individuelle ou globale est remplacé par une méthode exacte. Dans cette classe de coopération, diverses approches sont proposées. Ainsi dans [Cota, 1995] une approche basée sur la coopération entre AGs et B&B a été proposée. l'algorithme exacte est incorporé dans l'AG pour servir d'opérateur de recombinaison. L'opérateur résultant explore de manière intelligente les solutions potentielles issues de la recombinaison de deux parents, afin d'en extraire la plus intéressante grâce à un algorithme de B&B. Dans [Jahuira, 2002], différentes coopérations entre AGs et méthodes exactes appliquées au problème du voyageur de commerce, ont été proposées. L'opérateur génétique de recombinaison est remplacé par un algorithme de B&B, d'arbre de recouvrement minimal et de méthode de backtracking. De plus, la population initiale est générée à l'aide d'un arbre de recouvrement minimal généré de manière exacte. Dans [Kostikas, 2004], Kostikas et Fragakis proposent une hybridation méta/exacte dans laquelle un algorithme de programmation génétique (PG) est incorporé dans un algorithme de B&B basé sur un problème programmation linéaire mixte (PLM). L'architecture coopérative employée ici utilise la PG pour générer l'expression dédiée à la sélection des nœuds à explorer. L'algorithme de PG exploite les caractéristiques du PLM étudié pour faire évoluer la méthode de sélection de nœud. La méthode générée remplace alors l'opérateur de sélection par défaut de l'algorithme de B&B pour le reste de l'exécution. Dans [Bent, 2004], Bent et Hentanryck utilisent le principe de la recherche sur large voisinage faisant intervenir des méthodes exacte pour l'exploration du voisinage pour trouver la (les) meilleure(s) solution(s) dans un sous espace de l'espace de recherche global. Pour résoudre le problème du voyageur de commerce. Shaw [Shaw, 1998], utilise le même principe pour le problème de routage des véhicules.

3) la classe HRH, coopération de haut niveau en mode relais (Heigh-level Relay Hybrid). Dans cette classe de coopération, les heuristiques et/ou méthodes exactes hybridées conservent leur intégrité. Dans [Bent, 2004], Bent et Van Hentanryck proposent un algorithme hybride en deux phases pour le problème de tournées de véhicules avec fenêtre de temps. Dans un premier temps l'algorithme minimise le nombre de véhicules par un algorithme de recuit simulé. Puis le coût de chaque tournée est minimisé par une recherche exacte sur un large voisinage pouvant relocaliser un grand nombre de clients. Un autre exemple de ce type de coopération est proposé par Portmann [Portmann98] pour résoudre le problème de Flow-shop hybride. Kleipeis [Kleipeis, 2003], propose une coopération entre αB&B (pour les problèmes non linéaire) et AG hybridé avec un algorithme de Conformational Space Annealing (CSA) pour la prédiction de structure de protéine.

4) la classe HTH, coopération de haut niveau en mode teamwoke (Heigh-level teamwork Hybrid). Ce type de coopération est assez difficile à mettre en œuvre. En effet les deux approches ne permettent pas de résoudre les mêmes types de problèmes. Donc il parait indispensable Que les deux approches traitent des parties différentes du problème, tout en étant indépendantes l'une envers l'autre. Charbier [Charbier, 2002], propose une coopération entre recherche locale et algorithme de génération de colonnes pour le problème de tournées de véhicules. Les deux méthodes coopèrent en parallèle en se fournissant diverses informations.

2.2 Classification à plat

Les critères retenus pour cette classification sont :

a) L'approche de résolution : qui indique si l'approche générale de la méthode coopérative est *exacte* ou *approchée*. Burke [Burke, 2001] décrit une méthode coopérative méta/exacte approchée où la méthode exacte est incluse dans un algorithme de recherche locale pour explorer de manière exacte les régions prometteuses de l'espace de recherche. Dans l'approche proposée par Charbier [Charbier, 2002], malgré q'une recherche locale intervient dans un algorithme de B&C afin de générer de nouvelles colonnes permettant de définir des plans de coupes pertinents. L'approche globale reste exacte.

b) Domaine d'application : qui indique si l'approche générale de la méthode coopérative est *globale* (toutes les méthodes hybridées sont appliquées à la totalité de l'espace de recherche) ou *partielle*. Pour la coopération partielle, le problème à résoudre est décomposé en sous-problèmes, chacun ayant un espace de recherche propre. Ainsi chaque métaheuristique et méthode exacte de la coopération résout un sous-problème dans son propre espace de recherche. Une part importante des coopérations méta/exacte est partielle étant donné que l'espace exploré est trop large pour la méthode exacte seule. Mais des coopérations globales sont réalisées en vue d'accélérer une méthode exacte qui est déjà praticable sur les problèmes étudiés. Portmann [Portmann, 1998] propose une coopération où une heuristique calcule des solutions initiales dans le but d'offrir de bonnes bornes pour le lancement de la méthode exacte. Les deux méthodes coopératives travaillent donc sur le même espace de recherche. Palpant [Palpant, 2001] propose une coopération partielle, une méthode faisant cohabiter résolution exacte et recherche locale résolvant optimalement au cours d'itérations successives un certain nombre de sous-problèmes du problème d'optimisation combinatoire initial. Tout en intégrant des procédés de diversification et d'intensification dans la recherche des solutions. Des résultats très encourageants sur un ensemble de problèmes issus de la littérature sont obtenus. Maniezzo [Maniezzo, 1999] propose une autre approche partielle pour résoudre le problème d'affectation quadratique par un algorithme coopératif LRH à base de colonie de fourmis. En utilisant une borne inférieure du problème afin de rendre plus efficace l'algorithme. La borne inférieure est calculée en résolvant de manière exacte le problème linéaire en nombres réels associés pour le B&C&P. Nous n'avons pas trouvé de référence de coopération proposant une approche exacte de type partielle. Cela parait difficilement imaginable puisque pour résoudre de manière exacte un problème, la méthode exacte doit opérer dans tout l'espace de recherche. Par contre les coopérations méta/exacte globales sont généralement des approches exactes. Le cas le plus couramment rencontré dans la littérature est celui où la métaheuristique permet de fournir des bornes et/ou des solutions initiales pour débuter un algorithme de B&B.

c) Uniformité du problème traité : qui indique si l'approche générale de la méthode coopérative est *généraliste* ou *spécialiste*. Pour les coopérations généralistes, toutes les méthodes hybridées traite le même problème d'optimisation. A l'inverse, les coopérations spécialistes combinent des méthodes qui s'attaquent à des problèmes différents. La plupart des approches coopératives trouvées dans la littérature sont généralistes. Cependant l'algorithme de T'kindt [T'kindt, 2002] proposait une coopération de type partielle spécialiste pour résoudre un problème de Flow-shop biobjectif à deux machines. La méthode exacte (algorithme de Jhonson) optimise un objectif, tandis que la métaheuristique (colonie de Fourmis) optimise le second objectif dans un ordre lexicographique. La métaheuristique

n'étant appliquée que sur les solutions optimales trouvées par la méthode exacte pour le premier objectif.

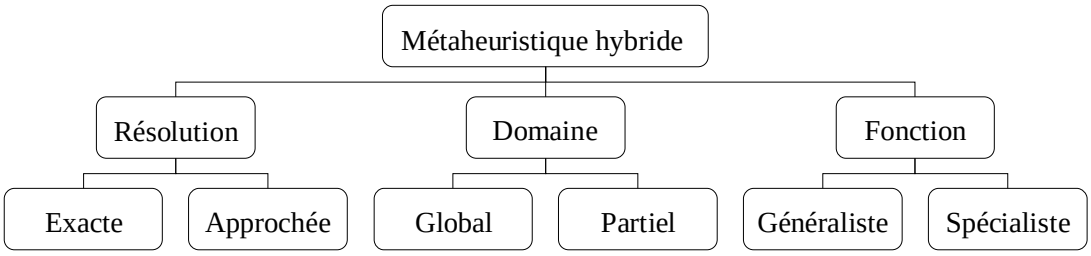


Figure 3.2 Structure de la taxonomie à plat

2.3 Hybridation collaborative

Les algorithmes échangent des informations, mais aucun algorithme ne fait pas partie de l'autre. Les algorithmes exactes et heuristiques peuvent être exécutés séquentiellement, entrelacé ou en parallèle.

a) Exécution séquentielle : Soit la méthode exacte est exécutée comme un prétraitement avant la métaheuristique, ou vice versa. Appliquer une méthode exacte et fournir des informations à la métaheuristique ou vice versa. Applegate [Applegate, 1998] a proposé une approche pour trouver des solutions proche de l'optimum pour résoudre le problème du voyageur de commerce. Un ensemble de diverses solutions est généré en exécutant plusieurs fois l'algorithme de la recherche local itérée. Et le problème est résolu à l'optimalité sur un ensemble restreint de graphe. Le résultat est supérieur à la meilleure solution de la recherche locale itérée. Klau et al. [Klau, 2004] ont suivi une idée semblable et combinent un algorithme mémétique avec la programmation linéaire en nombre entier pour résoudre approximativement le problème "prize-collecting Steiner tree". La structure algorithmique proposée consiste en trois parties : (1) une phase du prétraitement permet de réduire le graphe donné sans changer la structure de la solution optimale. (2) la partie centrale de l'approche est un algorithme mémétique (MA) basé sur un algorithme évolutionnaire d'état stable (steady-state evolutionary algorithm) et un sous-programme exact sur les problèmes d'arbres. (3) la solution de l'algorithme mémétique fournit un point de départ excellent pour une post-optimisation en résolvant une relaxation de programmation linéaire en nombre entier (ILP) pour trouver l'arborescence Steiner minimum dans un graphe dirigé. Les expériences sur des benchmark de la littérature montre que cette approche compare favorablement les résultats précédemment publiés. Par contre sur des moyens et grandes instances cette approche permet d'obtenir une réduction considérable de temps d'exécution.

b) Exécution entrelacée ou en parallèle : Idem à **HTH**

2.4 Hybridation intégrative

Une technique est un composant d'une autre technique. On distingue l'algorithme maître qui peut être exact ou métaheuristique, de l'algorithme esclave.

a) Algorithmes exacts incorporés dans des Métaheuristiques

– **Résolution exacte des problèmes relaxés** (Exactly Solving Relaxed Problems). Les algorithmes génétiques hybrides pour le problème du sac à dos multicontrainte de Chu et Beasley [Chu, 1998] et Raidl [Raidl, 1998].

– **Merging solutions**
C.Cotta et JM.Troya [2003] ont proposé une structure pour hybrider des algorithmes évolutionnaires avec le B&B. Cette structure est basée en utilisant B&B comme un opérateur enfoncé dans l'algorithme évolutionnaire. L'opérateur hybride explorera intelligemment le dynastique potentiel (enfants possibles) des solutions qui vont être recombinées.

– **Recherche exacte sur large Voisinages (**Exactly Searching Large Neighborhoods)
Les algorithmes de recherche sur large voisinage (large neighborhood search) sont des algorithmes de descente qui utilisent de larges voisinages pour améliorer l'efficacité de l'algorithme en faisant intervenir des méthodes exactes pour l'exploration du voisinage pour trouver la (les) meilleure(s) solution(s) dans un sous espace de l'espace de recherche global. Burke et al. [Bruke, 2001] présentent une métaheuristique de recherche locale à voisinage variable (VNS) pour le problème du voyageur de commerce asymétrique dans laquelle ils ont enfoncé un algorithme exacte dans la partie de la recherche locale, appelée HyperOpt pour chercher exhaustivement des régions prometteuses relativement grandes de l'espace de la solution.

– **Algorithmes exacts comme Décodeurs dans les algorithmes évolutionnaires**
Quelquefois les solutions candidates sont incomplètement représentées dans le chromosome et un algorithme exact est utilisé comme décodeur pour déterminer les parties manquantes. Staggemeier et al. [Staggemeier, 2002] par exemple, présentent un algorithme génétique hybride où la solution est décodée en résolvant un programme linéaire. L'approche est meilleure à la formulation MIP du problème résolu par CPLEX.

b) Métaheuristiques incorporées dans des algorithmes Exacts

– **Métaheuristiques pour Obtenir des Solutions titulaires et des bornes** (Metaheuristics for Obtaining Incumbent Solutions and Bounds): en général, les métaheuristiques sont souvent utilisées pour déterminer des solutions titulaires et des bornes dans les algorithmes B&B. Par exemple, Woodrurff [Woodrurff, 1999] décrit une stratégie de la sélection basée 'chunking' pour décider à chaque noeud de l'arbre B&B si une recherche taboue est appelée pour trouver une meilleure solution titulaire. La stratégie basée 'chunking' mesure une distance entre le noeud courant et les noeuds déjà explorés par la métaheuristique pour influencer la sélection vers les points distants. Les résultats indiquent l'utilisation de la métaheuristique améliore la performance du B&B.

– **Métaheuristiqes pour la Génération de Colonne et de Coupe** (Metaheuristics for Column and Cut Generation): dans les algorithmes B&C et B&P, la dynamique de séparation des plans de coupe et de génération des colonnes est faite au moyen d'heuristiques pour accélérer le processus de l'optimisation. Filho et Lorena [Filho, 2000]

ont appliqué une heuristique pour la génération de colonnes au problème de coloration de graphe. Ils décrivent les principes de leur algorithme génétique constructif et donnent une formulation de la génération de colonne du problème. L'AG est utilisé pour générer les colonnes initiales et résoudre le problème esclave (the weighted maximum independent set problem) à chaque itération. La génération de colonnes est exécutée aussi long que l'AG trouve des colonnes avec des coûts de réduction négative. Le problème maître est résolu en utilisant CPLEX. Des résultats encourageant ont été obtenus. Puchinger et Raidl [Puchinger, 2004] ont proposé une nouvelle programmation linéaire en nombre entier pour le "three-stage two-dimensional bin packing problem". Un algorithme de B&P a été développé dans lequel une exécution rapide de génération de colonne en appliquant une hiérarchie de quatre méthodes : (a) une heuristique gloutonne, (b) un algorithme évolutionnaire, (c) résoudre une forme restreinte du problème en utilisant CPLEX, et finalement (d) résoudre le problème complet en utilisant CPLEX. La combinaison de tous les quatre algorithmes dans une structure B&P obtient les meilleurs résultats en terme de la valeur moyenne de l'objectif, du temps moyen d'exécution, et le nombre d'instance résolu.

– **Métaheuristiqes comme Stratégie pour guider la Recherche Exacte (**Metaheuristics for Strategic Guidance of Exact Search) French et al. [French, 2001] présentent un GA/B&B pour résoudre le problème de satisfiabilité et des problèmes de l'IP. Leur algorithme hybride combine le B&B générique au MIP-solver XPRESS-MP avec un AG "steady state". Il commence en traversant l'arbre B&B. Pendant cette phase, les renseignements de noeuds sont rassemblés pour suggérer des chromosomes à être ajouté à la population initiale de l'AG. l'AG commence en utilisant la population initiale augmentée. Quand l'AG termine, sa solution la plus en bonne santé est passée et greffée sur l'arbre B&B. et le contrôle est donné au B&B. Les résultats rapportés sur des instances du problème MAX-SAT montrent que cette approche hybride donne de meilleures solutions que B&B ou le GA seul.

– **l'Esprit de Metaheuristiques appliqué aux méthodes exactes**

Quelques approches essayent d'apporter l'esprit de recherche locale dans B&B. L'idée principale est de chercher en premier, quelques voisinages de solutions titulaire plus intensivement avant de tourner à une stratégie classique de la sélection des nœuds. B&B lui-même est utilisé pour faire de la recherche locale. Fischetti et Lodi [Fischetti, 2003] ont introduit le branchement local, une approche exacte combinant l'esprit d'une métaheuristique de la recherche local avec un MIP (CPLEX). Ils considèrent MIPs avec 0-1 variables. L'idée est de résoudre itérativement un sous-problème local qui correspond à un voisinage k-Opt classique qui utilise le MIP-(CPLEX). Cela est accompli en introduisant une contrainte du branchement locale basée sur une solution titulaire $\bar{x}$ qui découpent l'espace de recherche dans le k-Opt voisinage et le reste: $\Delta(x,\bar{x}) \le k$ et $\Delta(x,\bar{x}) \ge k+1$ respectivement, avec Δ est la distance de Hamming des 0-1 variables. Le premier sous-problème est résolu, et si une solution améliorée puisse être trouvée, un nouveau sous-problème est imaginé et est résolu; cela est répété aussi long qu'une solution améliorée est trouvée. Si le processus s'arrête, le reste du problème est résolu d'une manière standard. Ce mécanisme de base est étendu en introduisant le temps limite, en modifiant automatiquement la dimension de voisinage k en ajoutant une stratégies de la diversification pour améliorer la performance. Les résultats rapportés sont prometteurs.

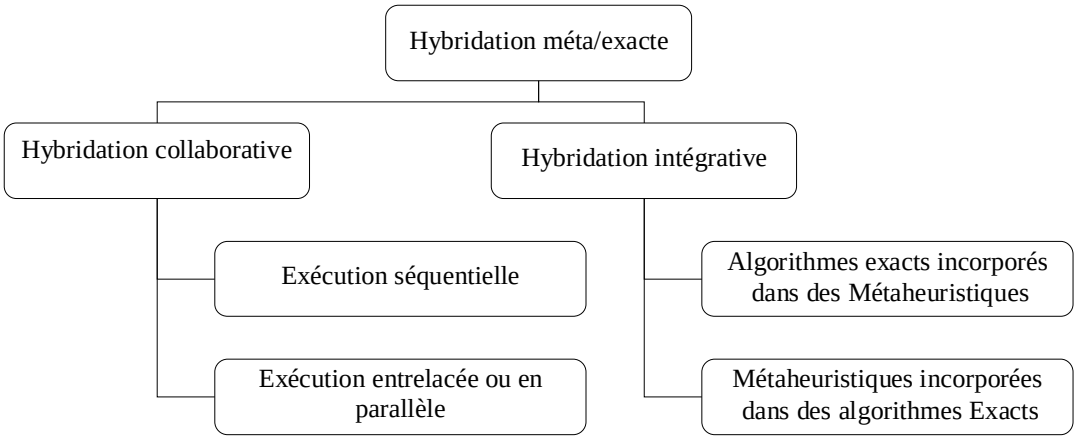


Figure 3.3 Majeur classification de coopération méta/exacte

3 Coopération méta/exacte en optimisation multiobjectif

En multiobjectif peu de travaux méta/exacte ont été réalisés. On peut citer les travaux de M.Basseur dans sa thèse de doctorat [Basseur, 2005] qui présente trois méthodes réalisées entre un AG Adaptatif et la méthode à deux phases :

- La première est une approche de résolution exacte de type HRH. La métaheuristique est lancée dans un premier temps, puis les valeurs trouvées sont utilisées pour améliorer l'initialisation de la méthode exacte. Cependant, l'apport n'est pas suffisant pour permettre de résoudre exactement des problèmes de plus grandes tailles.
- Les deux autres sont approchées de type HTH, l'une explorant des voisinages larges, l'autre s'appuyant sur le partitionnement des solutions. les recherches de large voisinage sont effectuées sur chaque individu non dominé fourni par la métaheuristique (très gourmande en temps de calculs). Dans l'optimisation par partitionnement, une population initiale est générée par la métaheuristique. Puis on optimise partition par partition l'ensemble des solutions par la méthode exacte.
- Dans sa thèse, Nicolas Jozefowiez [Jozefowiez, 2004] propose une méthode coopérative pour la résolution d'un problème de tournées (le problème de la tournée couvrante) bi-objectif. Pour cela un algorithme génétique propose une approximation puis, un algorithme de séparation et coupes (Branch and Cut) est utilisé pour résoudre optimalement des sous-problèmes suivant l'un des objectifs.

4 Conclusion

Nous avons présenté brièvement les différentes classifications des approches de coopération méta/exacte que nous avons trouvé dans la littérature. Avec ce recensement, on remarque que quelques voies de coopérations peuvent encore être explorées ou approfondies. De plus, il ressort que très peu de travaux traitent des coopérations dédiées à l'optimisation multiobjectif.

Notre schéma de coopération méta/exacte qui sera développé dans le chapitre suivant, se propose de combiner résolution exacte de sous-problèmes et résolution heuristique du problème complet. La méthode développée notée (MA|PM)Ä(B&B) est une hybridation intégrative, de classe HRH, et de model à plat (approchée, partiel, généraliste). L'algorithme principal est le MA|PM : algorithme mémétique avec gestion de la population, renforcé par un algorithme de type Branch & Bound.

Chapitre 4

Un schéma de coopération méta/exacte : application au problème du sac à dos

Dans ce chapitre,

Sommaire

1 Introduction

L'intérêt des problèmes d'optimisation combinatoire (en particulier les problèmes multiobjectifs) dans l'économie, la science, la médecine, l'industrie, l'ingénierie, etc. a provoqué une intensification des études portant sur ce type de problèmes. Les approches exactes sont très vite limitées, et des métaheuristiques dédiées à la résolution de ces problèmes ont été mises au point, mais la difficulté de ce type de problèmes encourage l'utilisation de méthodes d'optimisation coopératives.

Pour répondre à l'objectif de concevoir un algorithme efficace, en temps de calculs (raisonnable), en qualité de solutions produites (l'intensification et la diversification) et permettant de traiter des problèmes de grande taille ; nous avons adopté la méthode hybride méta/exacte. L'idée principale est d'hybrider résolution exacte de sous-problèmes et résolution heuristique du problème complet, en combinant des méthodes au pouvoir d'intensification et des méthodes qui possèdent des facultés de diversification. Ainsi, notre approche comporte : une méthode de ranking qui est un facteur de convergence, une méthode de diversification basée sur le crowding, une méthode d'élitisme permettant une meilleure intensification de la recherche, et un nouveau schéma de coopération méta/exacte réalisé entre une métaheuristique avancée MA/PM : Memetic Algorithm with Population Management, et un algorithme exact de type Branch & Bound au pouvoir de recherche absolu. Pour l'évaluation des performances nous avons choisi le problème académique du sac à dos sous ses différentes formes : mono-objectif, multidimensionnel et multiobjectif. Nous avons réalisé nos expérimentations sur des benchmarks bien connus dans la littérature. La version mono-objectif a été testée sur les mêmes instances que les travaux de K.H.Han et J.H.Kim [Han, 2000] obtenus par leur algorithme génétique quantique. Dans la version multidimensionnel, les tests ont été effectués sur un nouvel ensemble d'instances référencé par OR-library à l'adresse « hces.bus.olemiss.edu/tools.html ». Enfin, notre centre d'intérêt est la version multiobjectif basée sur le concept d'optimum Pareto et qui a été testée sur des problèmes présentés par Zitzler à l'adresse « tik.ee.ethz.ch/~zitzler/testdata.html ». Les résultats obtenus mettent en évidence le bon comportement de notre méthode, et la comparaison entre la métaheuristique MA|PM et notre méthode (la même métaheuristique combinée à un B&B), montre l'efficacité de l'approche méta/exacte.

2 Problème du sac à dos

Le problème du sac à dos (*knapsack* problem) est un problème académique d'optimisation combinatoire appartenant à la classe des problèmes NP-difficile. On le retrouve sous de nombreuses formes, mono-objectif KP, multidimensionnel [M]KP et multiobjectif MOKP. Le fait qu'on le rencontre dans des domaines d'application aussi différents que l'économie (la répartition de budgets), les transports (chargement de cargaison) et l'informatique répartie (allocation de ressources), etc. lui confère un grand intérêt pratique. Le principe consiste à sélectionner un sous-ensemble d'objets maximisant une fonction dite objectif de manière à transporter la plus grande valeur. Son expression Formelle est la suivante :

$$\begin{cases} \max Z^j(x) = \sum_{i=1}^{n} v_i^j x_i \quad j=1,\ldots, p\ ; \\ \sum_{i=1}^{n} w_i^l x_i \le W^l \quad l=1,\ldots k\ ; w_i^l \ge 0\ ; \\ x_i \in \{0,1\} \end{cases}$$

- Un sac de capacité ou volume scalaire W si $k=1$ ou vectoriel W^l si $k>1$ à ne pas dépasser.
- Une fonction objectif scalaire Z si $p=1$ ou vectoriel Z^j si $p>1$ à optimiser (maximiser).
- n : Nombre d'objets où chaque objet i possède, un poids w_i si $k=1$ ou un vecteur de poids w_i^l si $k>1$. Et un vecteur de valeur v_i^j par rapport au critère *j*.
- On définit pour chaque objet i une variable de décision binaire x_i. Si l'objet *i* est retenu alors $x_i = 1$ sinon $x_i = 0$.

k \ p	=1	>1
=1	KP (Mono-objectif)	MOKP (Multiobjectif)
>1	[M]KP (Multidimensionnel)	MO[M]KP (Multidimensionnel Multiobjectif)

L'espace de recherche est composé de toutes les configurations données par les vecteurs binaires $(x_1, x_2,\ldots, x_n)$ vérifiant toutes les contraintes de sac à dos**.**

3 Description de la méthode

Notre approche de résolution combinatoire se propose de combiner résolution exacte de sous-problème et résolution heuristique du problème complet. La méthode développée notée (MA|PM)⊗(B&B) est une hybridation intégrative, de classe HRH, et de model à plat (approchée, partiel, généraliste). L'algorithme principal est le *MA*|*PM* : algorithme mémétique avec gestion de la population, renforcé par un algorithme de type *Branch & Bound***.** Elle comporte une méthode *d'élitisme.* La version multiobjectif est basée sur le concept *d'optimum Pareto* et comporte une méthode de *ranking* et une méthode de *crowding*.

3.1 Schéma d'hybridation méta/exacte

La méthode (MA|PM)⊗(B&B) est composée d'un algorithme mémétique MA, d'un module de gestion de la diversité de la population PM et d'un algorithme exact de type B&B. l'idée d'un algorithme mémétique est de rendre un algorithme génétique GA plus agressif par l'ajout d'une méthode de recherche locale LS. L'algorithme génétique utilisé est de type SSGA (Steady State Genetic Algorithm) ce modèle permet d'alléger la phase d'évaluation en appliquant une seule reproduction par génération. L'algorithme de recherche locale utilisé est la plus grande descente (Deepest Descent) DD.

3.1.1 Algorithme mémétique avec gestion de la population (MA|PM)

Conçu dans le but de respecter l'équilibre souhaité entre l'intensification et la diversification. Le fonctionnement est basé sur un ***algorithme génétique*** et s'appuie sur trois principes importants :

une population de taille réduite, un opérateur de ***recherche locale*** et une ***gestion active de la diversité de la population*** tout au long de la recherche. Il combine les avantage :

- du scatter search concernant la mesure de la diversité des solutions. A la différence du scatter search, la diversité est contrôlée précisément.
- D'un algorithme mémétique concernant l'application d'une recherche locale à toutes les nouvelles solutions.
- Du recuit simulé concernant l'évolution du paramètre de diversité Δ comme dans les schémas de refroidissement de température qui permet à tout moment d'augmenter ou de réduire la diversité des individus dans la population.
- En plus elle maintient une population de petite taille et diversifiée.

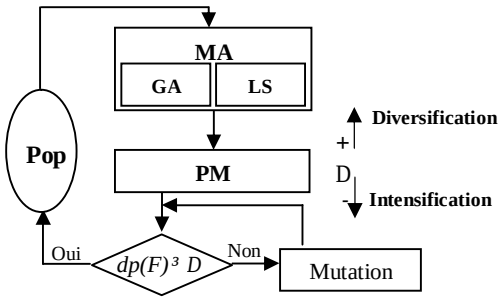


Figure 4.1 La métaheuristique MA|PM

Les algorithmes évolutionnaires, et particulièrement les algorithmes génétiques, sont très adaptés à l'optimisation multiobjectif. En remplaçant durant l'étape de sélection la comparaison d'objectifs scalaires, par la notion de dominance au sens Pareto (ranking), et sans presque rien changer au reste de l'algorithme MA|PM, on arrive à l'adapter au cas multiobjectif.

3.1.1.1 Algorithme génétique

Un AG est un algorithme stochastique itératif qui opère sur des ensembles de points codés, aux moyens d'opérateurs simples, simulant l'évolution naturelle et génétique pour résoudre des problèmes complexes. Deux types d'algorithme génétique : ***Algorithme Génétique Générationnel*** (Simple Genetic Algorithm SGA) et ***La stratégie de remplacement stationnaire*** (Steady State Genetic Algorithm SSGA). Dans le SGA, une nouvelle population en entier est générée par itération. Le nombre de descendants produits est donc égal au nombre d'individus parents. Dans le SSGA, un ou deux individus qui sont générés à la fois sont ajoutés à la population. Cette stratégie a l'avantage d'alléger la phase d'évaluation. Notre algorithme mémétique est basé sur un SSGA.

- ***Fonctionnement*** : Faire évoluer, à partir d'un tirage initial, un ensemble de points de l'espace de recherche vers l'optimum du problème. Avec des opérations extrêmement simples effectuées sur les solutions codées en les combinant entre elles pour en former de nouvelles. u Au fil des itérations la population va se centrer d'elle-même autour d'une solution qui s'approche de l'optimum en un temps acceptable. C'est en effet cette légèreté de mise en œuvre qui fait la puissance des AGs. La combinaison génétique sur laquelle repose la meilleure solution, est dispersée chez plusieurs individus. Ce n'est que par l'association de ces combinaisons génétiques entre individus que l'on souhaite approcher au mieux cette solution. La probabilité de construire une très bonne solution est fortement augmentée si on recombine des individus de bonne qualité. Mais, il ne faut pas complètement écarter les individus moins

bons, car ils peuvent posséder un matériel génétique intéressant. Les AGs ont montré leur efficacité pratique bien avant que les résultats de convergence théorique ne soient établis.

– ***les éléments d'un AG*** :

- Codage
- Fonction fitness
- Processus d'évolution
  - Sélection
  - Croisement
  - Mutation
  - Remplacement
- Paramétrage

– ***Codage des solutions*** **:** Chaque solution possible (Phénotype) est codée sous la forme d'un ensemble de chromosomes appelé individu (Génotype). Un chromosome est une chaîne ordonnée de gènes par analogie à la molécule d'ADN. Il faut donc définir : l'alphabet des gènes (en générale {0, 1}) et la structure du chromosome.

– ***La fitness*** : mesure la qualité de la solution et donc la capacité de survie. La fonction fitness est le plus souvent la valeur de la fonction objectif de la solution associée à l'individu. elle est utilisée pour converger le processus vers l'optimum.

– ***Processus d'évolution*** : Construction successive de nouvelle génération

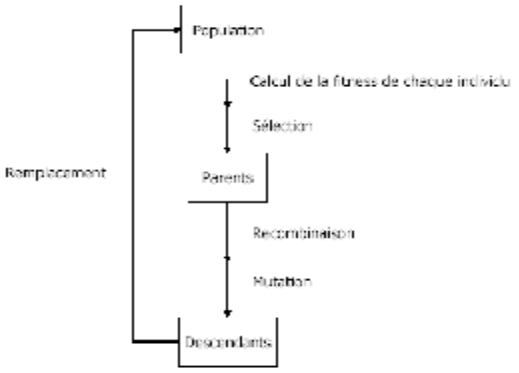
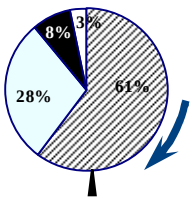


Figure 4.2 Schéma du processus d'évolution d'un AG de base

– ***Sélection*** : Identifier les individus qui doivent se reproduire sur la base de leur fonction d'adaptation. La roulette, le tournoi, le rang …

- **La roulette**

$$prob(j) = \frac{fitness(j)}{\sum_{j=1}^{n} fitness(j)}$$



Un bon individu a une fitness élevée, donc un large secteur de roulette, et alors il aura plus de chance d'être sélectionné. Les meilleurs individus se reproduisent rapidement (risque de convergence prématurée). Pas de sélection objective si les valeurs des fitness son proches.

§ Le tournoi

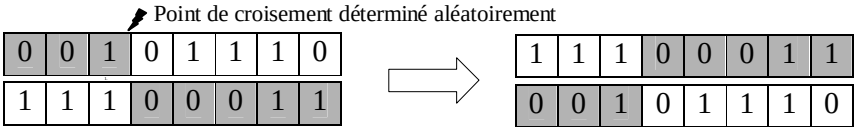
On tire aléatoirement ***k*** individus qui vont entrer en compétition (fitness). Le meilleur accède à la génération intermédiaire et on répète l'opération jusqu'à remplir la génération intermédiaire (*N/2* individus). Dans cette stratégie favorise le clonage des gagnants, ce qui assure la pérennité de leurs gènes. Le nombre de clones a une influence sur la pression de sélection.

§ Le rang

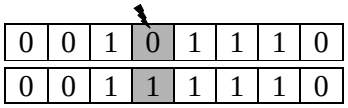
On trie la population par fitness. À chaque chromosome est associé un rang en fonction de sa position, de 1 à *N* (du plus mauvais au meilleur). Ensuite le principe est le même que pour la roulette mais les proportions sont liées au rang. Dans cette stratégie tous les individus gardent une bonne chance d'être sélectionné (ce qui évite les convergences prématurées). Par contre la convergence est plus lente car les bons individus se distinguent moins bien du lot.

– ***Croisement*** : Mélange du matériel génétique, de deux individus ou plus (croisement multi-parents), avec une probabilité P_c.

§ Croisement à un point : est une opération qui permet de créer de nouvelles solutions à partir de deux individus.



– ***Mutation*** : Perturbation du matériel génétique d'un individu (sur un gène), avec une probabilité $P_{m,}$ pour éviter le blocage dans un optimum local permettant d'ajouter de la diversité. Elle s'applique à un gène et a pour effet de changer sa valeur (dans le cas d'un bit on inverser sa valeur)



– ***Remplacement*** : L'individu victime à remplacer par une nouvelle solution générée est choisi selon une stratégie.

§ aléatoirement

§ celui qui a la plus petite fonction d'adaptation

§ Les enfants remplacent leurs parents

– ***Paramétrage*** : les éléments précédents dépendent de plusieurs paramètres qui sont fixés à l'avance et dont dépendent fortement les performances (la vitesse de convergence et la qualité de la solution obtenue) de l'algorithme, Il faut donc bien les paramétrer : *Taille de population, critère d'arrêt, pression de sélection, la population initiale doit être hétérogène, etc.*

Structure d'un SSGA

t ← 0, Initialiser la population de taille N (aléatoire).
Evaluer les individus de la population. (fitness)
Tant que le critère d'arrêt n'est pas atteint faire
    Sélection deux individus à coupler.
    Reproduction (croisement, mutation).
    Evaluer les deux enfants (fitness)
    Remplacement,
    t ← t+1
Fin.

Le concept de population est très approprié dans le contexte de l'optimisation multiobjectif. En effet, comme le front Pareto optimal est dans la plupart des cas, un ensemble de solutions, chaque individu de la population peut espérer devenir une solution particulière du front Pareto optimal. En conséquence, la population dans son intégralité devient une approximation de l'ensemble Pareto optimal [Barichard, 2004]. Les algorithmes évolutionnaires, et particulièrement les algorithmes génétiques, sont très adaptés à l'optimisation multiobjectif. En remplaçant durant l'étape de sélection la comparaison d'objectifs scalaires, par la notion de dominance au sens Pareto (ranking), et sans presque rien changer au reste de l'algorithme, on arrive à l'adapter au cas multiobjectif.

Structure d'un SSGA multiobjectif

t ← 0, Initialiser la population de taille N (aléatoire).
Ranking
Evaluer les individus de la population. (rang, dominance)
Tant que le critère d'arrêt n'est pas atteint faire
    Sélection deux individus à coupler.
    Reproduction (croisement, mutation).
    Evaluer les deux enfants (rang, dominance)
    Remplacement,
    t ← t+1
Fin.

L'AG présente l'avantage de mettre en place un parallélisme intrinsèque et d'explorer ainsi un espace très vaste de solutions. L'une des faiblesses d'un algorithme génétique basique est souvent la faible vitesse de convergence. Alors pour le rendre plus efficace on le combine avec d'autres méthodes de recherche locale pour mieux exploiter les bonnes solutions trouvées. Cette hybridation connue sous le nom d'algorithme mémétique, fut introduite par Moscato en 87.

3.1.1.2 Algorithme de recherche locale « la plus grande descente »

Fonctionne dans une structure de voisinage en se déplaçant d'une solution vers une autre. A partir d'une solution existante, chercher la meilleure dans le voisinage, qui devient la nouvelle solution. Et on recommence le processus jusqu'à ce qu'il n'y ait plus aucune solution améliorante. Pour la version multiobjectif, nous attachons à la notion de voisinage, la notion de Pareto localement optimale : une solution qui n'est dominée par aucune solution de son voisinage.

Le voisinage est défini de la manière suivante : soient x_1, x_2 deux solution, x_1 et x_2 sont voisin s'ils diffèrent par la valeur d'une seule variable. Formellement *distanceHamming*(x_1,x_2)=1. Il en résulte qu'à partir d'une configuration x, il est possible d'obtenir une configuration voisine x' en ajoutant ou en enlevant un objet i (x_i : 0→1 ou 1→0) de x tel que x' reste réalisable. Un autre aspect important est la taille du voisinage (le nombre de variables différent entre deux solutions).

Exemple : | 1 | 0 | 0 | 1 | 0 | 1 | est un voisin de | 0 | 0 | 0 | 1 | 0 | 1 |

| 1 | 0 | 0 | 1 | 0 | 1 | est un voisin d'ordre 3 de | 0 | 0 | 0 | 1 | 0 | 1 |

Algorithme Deepest descent (X)

Répéter
  Trouver une solution X'∈ N (X) / f (X') ≥ f (X''), ∀X''∈ N (X) (*recherche de voisinage
  Si f (X') > f (X) Alors
    X' ß X
  Fin Si
Jusqu'à f (X') ≤ f (X), ∀X' ∈ N (X)

Algorithme Deepest descent Multiobjective (X)

Répéter
  Trouver une solution X'∈ N (X) / X' Domine X (*recherche de voisinage
  Si existe
    X' ß X
  Fin Si
Jusqu'à not existe

3.1.1.3 Gestion de la population (PM)

Un des points importants, c'est la mesure contrôlée de la diversité des solutions (comme dans le *scatter search*), qui permet en fonction de l'ensemble des individus de la population de plus ou moins diversifier et de plus ou moins intensifier la recherche. A la différence du *scatter search*, la diversité est contrôlée précisément. Cette diversité est mesurée par une fonction de distance calculée dans l'espace décisionnel entre les individus. $dp(F)=min\ d(F,X_i)$, dp: distance à la population, $Xi \in$ population, F: nouvel individu et d: la distance de Hamming. La gestion de la population contrôle la diversité par la manipulation d'un paramètre de diversité (Δ), fixant le niveau de dissemblance des solutions entre elles. Cette gestion permet à une nouvelle solution F d'être ajoutée à la population si sa distance à la population est supérieure ou égale à la valeur du paramètre de la diversité $(dp(F) \geq \Delta)$. Autrement la nouvelle solution est mutée jusqu'à satisfaction. La manipulation du paramètre de diversité (Δ) peut être interprétée dans les deux sens. Si on diminue (Δ), on autorise l'intégration de solutions moins diverses, donc on intensifie la recherche, dans le cas contraire on force les individus à être de plus en plus éloignés entre eux, et on diversifie donc la recherche.

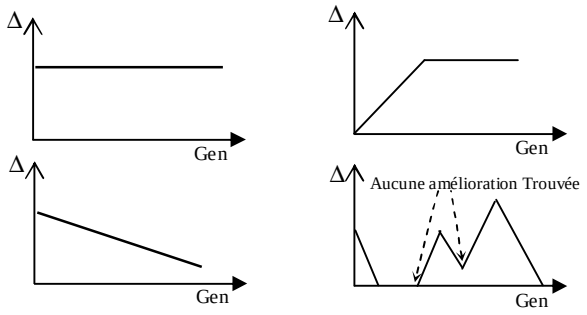


Figure 4.3 Stratégies de gestion de la population

3.1.2 Algorithme exact de type Branch & Bound

La méthode par séparation et évaluation (Branch & Bound) est une méthode générique de résolution exacte de problèmes d'optimisation. C'est une méthode constructive basée sur une recherche arborescente. L'idée de base est de construire un arbre qui exprime implicitement toute les solutions du problème, en se basant sur deux concepts : la séparation qui consiste à diviser l'espace des solutions en sous-problèmes pour les optimiser chacun individuellement, et l'évaluation basée sur la notion de bornes, calculées et comparées (relation de dominance dans le cas multiobjectif) à une solution déjà trouvée permet d'éliminer des branches de l'arborescence sans les parcourir. Le problème initial est la racine correspondant à une solution partielle vide, les feuilles correspondent à des solutions réalisables. Les autres nœuds correspondent à des solutions partielles. Une borne est attribuée à chaque nœud. Durant l'exploration, si une branche est moins bonne que (dominée par) la solution courante, on arrête de l'explorer. Si une solution est meilleure que la solution courante, elle la remplace. Tout sous-problème non réalisable sera coupé.

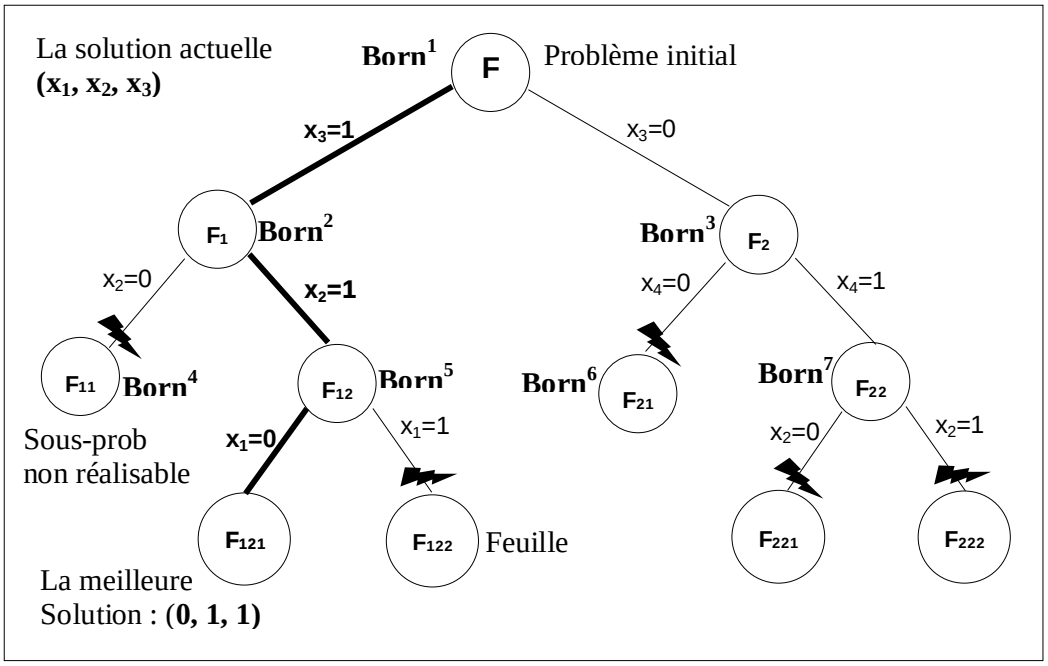


Figure 4.4 illustration de la construction de l'arbre de recherche d'un algorithme de type B&B

En plus, dans le cas multiobjectif, toutes les solutions non comparables à la solution courante, sont stockées dans un ensemble (PO). Si une solution domine la solution courante, elle la remplace et l'ensemble (PO) est mis à jour.

3.2 Elitisme

Dans le cas d'un problème mono-objectif, l'élitisme consiste à recopier dans la génération suivante la meilleure solution. Dans le cas multiobjectif cela consiste à maintenir une population externe, qui permet d'archiver le meilleur ensemble des points non dominés découverts jusqu'ici. Cet ensemble est mis à jour continuellement pendant la recherche, et participe avec une certaine probabilité à l'étape de sélection. Cette méthode permet ainsi une intensification de la recherche. Actuellement, les algorithmes élitistes obtiennent de meilleurs résultats sur un grand nombre de problèmes multiobjectifs [16 82.doc]**.**

3.3 Ranking (mécanisme de sélection Pareto)

L'utilisation d'un algorithme évolutionnaire dans un contexte multiobjectif nécessite de pouvoir associer une valeur scalaire unique (la fitness), au vecteur des objectifs. Ce principe appelé ranking, consiste à classer les individus en leur donnant un rang. La valeur d'adaptation est alors attribuée à chaque individu en se basant sur son rang. Cette fitness sera utilisée dans l'étape de sélection de l'algorithme (c'est ce mécanisme de sélection Pareto qui offre une alternative élégante et efficace aux algorithmes évolutionnaires de s'adapter facilement au cas multiobjectif). Le ranking NSGA utilisé dans notre méthode consiste à affecter le rang 1 à tous les individus non dominés de la population. Ces individus sont ensuite enlevés de la population, et l'ensemble suivant d'individus non dominés est identifié et on leur attribue le rang 2. Ce processus est réitéré jusqu'à ce que tous les individus de la population aient un rang. Cet ordre dépend de la notion de dominance (*figure* 4.5) et donc directement de l'optimalité Pareto. La méthode de ranking permet ainsi de converger vers les solutions Pareto optimales.

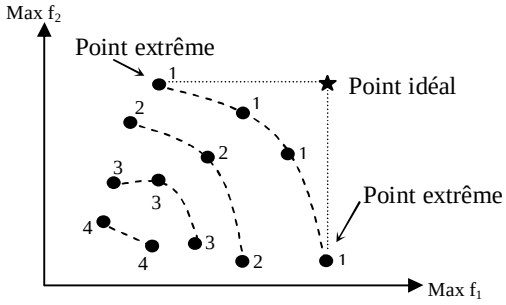


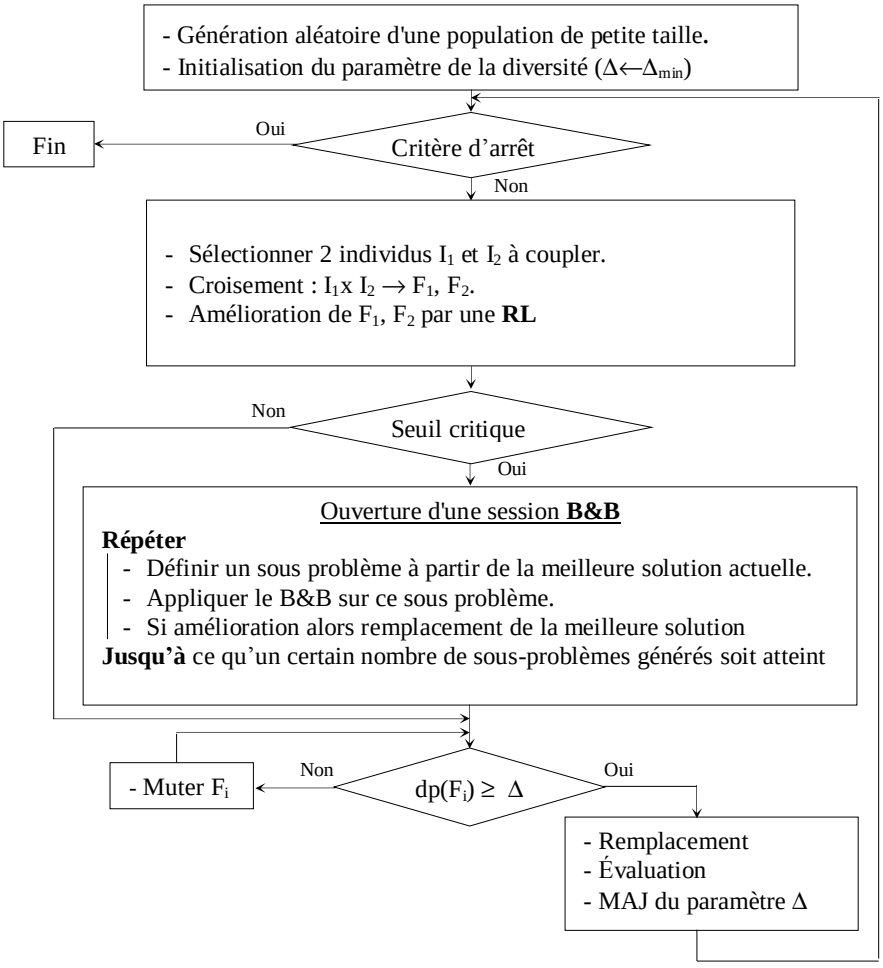
Figure 4.5. *Ranking NSGA (classement des individus par fronts)*

3.4 Crowding

Le principe du crowding utilisé est celui de Holland [Holland, 1975] : l'individu généré remplace dans la population l'individu qui lui est le plus semblable.

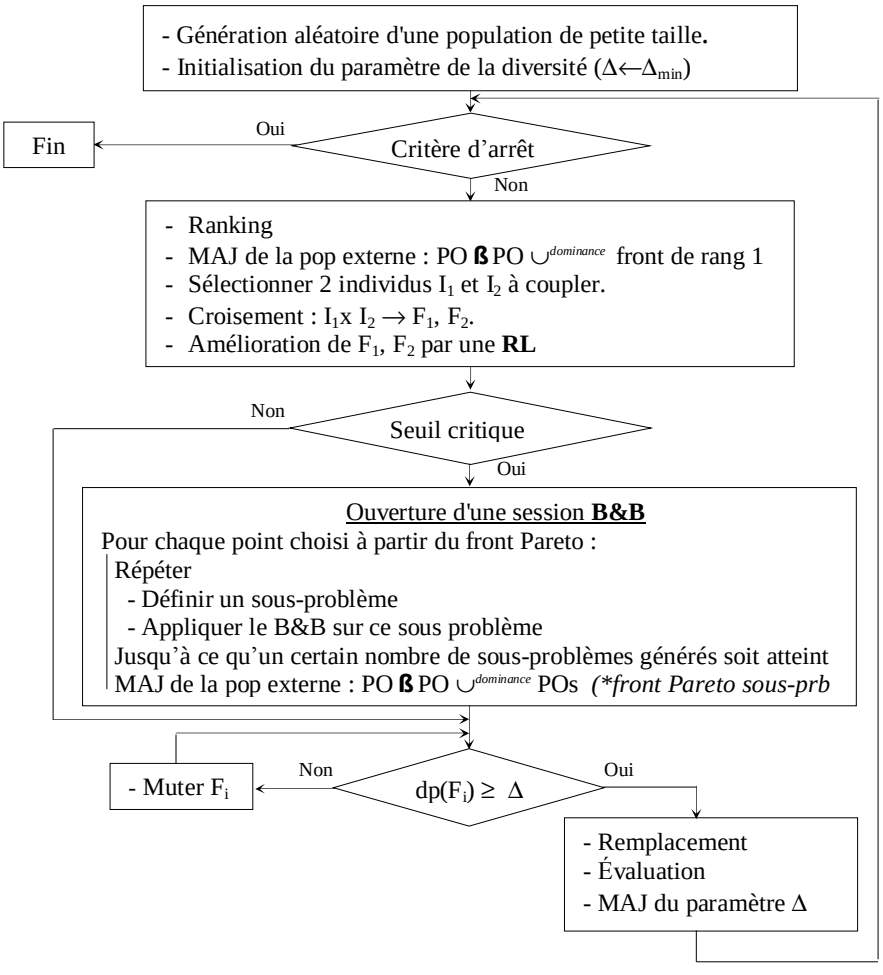
4 Conception

Le *B&B* cohabite avec la méthode de recherche locale au sein de *MA|PM*. Cette cohabitation conçoit une méthode de recherche locale méta/exacte basée sur la notion de voisinage. La recherche locale, *la plus grande descente* est appliquée à chaque nouvelle solution produite. Le *B&B* est appliqué au voisinage d'ordre [10 à 20] autour de la meilleure solution actuelle (cas mono-objectif). Dans le cas multiobjectif Le *B&B* est appliqué au voisinage d'ordre [10 à 20] sur des solutions du front Pareto actuel. L'algorithme maître est le *MA|PM*, dont le fonctionnement est basé sur un *SSGA* qui a l'avantage d'alléger la phase d'évaluation en appliquant une seule reproduction par génération. Le rôle principal du module *PM* (gestion de la population) est de contrôler la diversité le long de la recherche par la manipulation du paramètre ($\Delta \in [\Delta_{min}, \Delta_{max}]$), qui permet en fonction de l'évolution de la population de plus ou moins diversifier et de plus ou moins intensifier la recherche. Cette gestion permet de réaliser une caractéristique importante : c'est de pouvoir réduire la taille de la population (une petite population sans gestion de la diversité, converge prématurément d'une manière très rapide vers une population homogène).



mono-objectif

Initialement ($\Delta \leftarrow \Delta_{min}$), ce qui permet d'entamer une phase d'intensification. A partir de cette population, deux individus sont choisis selon une stratégie de sélection et croisés pour produire deux descendants qui sont améliorés par recherche locale, ensuite ajoutés à la population selon une stratégie de remplacement s'ils satisferont au critère de la diversité ($dp(F_i) \geq \Delta$), sinon ils seront mutés (éloignés) jusqu'à satisfaction. Si aucune amélioration n'a été trouvée durant un certain nombre de générations en phase d'intensification, alors une phase de diversification est exécutée pendant un certain nombre de générations, pour réorienter la recherche vers d'autres zones, en faisant varier graduellement (Δ). A chaque amélioration (mesurée par la procédure d'évaluation), la population est récompensée en réinitialisant la valeur de Δ à Δ_{min} pour mieux exploiter son entourage. Après chaque phase de diversification, on entame une phase d'intensification pour mieux exploiter la nouvelle zone visitée. Cette oscillation entre intensification et diversification en fonction de l'évolution de la population est exécutée jusqu'à ce que le critère d'arrêt de l'algorithme soit atteint. Ce rythme est orchestré par la procédure de mise à jour du paramètre (Δ).



jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj

En multiobjectif l'amélioration est mesurée par le point idéal (voir *figure* 4.5) et la cardinalité de l'ensemble des solutions de la population externe (le front Pareto).

Evaluation de l'amélioration en multiobjectif

Si point_idéal (front actuel) ***domine*** point_idéal (front précédent)
Alors amélioration ß True *(* amélioration de la convergence*
Sinon **Si** card (front actuel) > card (front précédent)
Alors amélioration ß True *(* amélioration de la diversité*
Sinon amélioration ß False
Finsi
Finsi

Procédure d'évaluation

A chaque génération
Si phase d'intensification
Alors évaluation de l'amélioration
Si pas d'amélioration
Alors C∅_sans_amelioration ß C∅_sans_amelioration + 1
Sinon C∅_sans_amelioration ß 0
Δ ß Δ_{min} *(* récompenser la population*
Finsi
Finsi

Procédure MAJ du paramètre D

Si seuil de diversification est atteint
Alors **Selon** la phase actuelle faire
(* *Phase diversification* $\Delta \neq \Delta_{min}$
Si pourcentage de la diversification est atteint
Alors
Sauvegarder Δ
Basculer vers l'intensification (Δ ß Δ_{min})
Finsi
(* *Phase d'intensification* $\Delta = \Delta_{min}$
Restaurer la valeur de Δ
Δ ß $\Delta \pm 1$ *(* Incr de D_{min} à D_{max} ou Décr de D_{max} à D_{min}*
Finsi

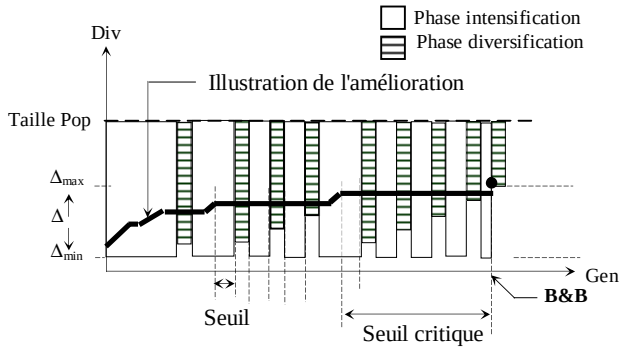


Figure 4.6 Illustration MAJ du paramètre D (Nous avons conçu une stratégie adaptée à notre méthode)

Ouverture d'une session B&B

La nature coûteuse d'une recherche exacte, impose l'intervention du B&B après un certain nombre de générations sans amélioration *critique* :

- Cas mono-objectif (un blocage dans un maximum local) : pour mieux intensifier la recherche autour de la meilleure solution actuelle. En explorant exactement quelques voisinages d'ordre [10 à 20] de cette solution, en générant aléatoirement plusieurs sous-problèmes à partir de cette dernière. La définition d'un sous-problème consiste à choisir d'une manière aléatoire les variables à passer au sous-problème et les autres variables sont fixées à leurs valeurs (*Voir exemple ci-dessous*).

Procédure B&B (*sous-problème, borne :* tous les objet du sous-problème sont retenus*, pile d'exploration***)**

```
Répéter
  Si nsœud-visité est intermédiaire
  Alors Si fitness de la meilleure solution ≥ borne ou estimation de la branche conduit à une solution non-réalisable
        Alors couper la branche (dépiler)
        Sinon SÉPARATION et ÉVALUATION
        Finsi
  Sinon Si le nœud-visité est une feuille qui correspond à une solution réalisable
        Alors Si la borne de la feuille > la fitness de la meilleure solution
              Alors amélioration_exacte ß true
              Finsi
        Finsi
  Finsi
Jusqu'à pile-vide
```

- Cas multiobjectif (un blocage dans un front Pareto local) : pour mieux intensifier la recherche sur ce front, en générant aléatoirement plusieurs sous-problèmes de taille fixée [10 à 20] à partir des points choisis du front Pareto. Nous avons choisi les points extrêmes du front Pareto (les points possédant les meilleures solutions pour chaque composant de la fonction objectif, voir *figure 4.5*) et un autre point choisi aléatoirement parmi le front Pareto. Ce choix permet de garder une meilleure allure du front.

Procédure B&B (*sous-problème, borne :* tous les objet du sous-problème sont retenus, *pile d'exploration***)**

Répéter

Si nsœud-visité est intermédiaire

Alors Si point-élu ***domine*** la borne ***ou*** l'estimation de la branche conduit à une solution non-réalisable

Alors couper la branche (dépiler)

Sinon SÉPARATION et ÉVALUATION

Finsi

Sinon Si le nœud-visité est une feuille qui correspond à une solution réalisable

Alors 3 cas envisageables

1 : cette configuration est incomparable au point-élu **Alors** MAJ *POs (* **front Pareto du sous-problème***

2 : cette configuration domine le point-élu **Alors** point-élu ← cette configuration, MAJ *POs*.

3 : cette configuration est dominée par le point-élu **Alors** ne fait rien.

Finsi

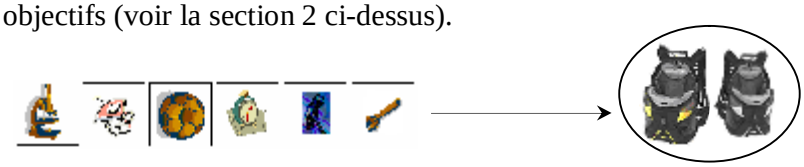
Jusqu'à pile-vide

Le résultat de cette recherche est soit un ensemble de solutions dominant le point choisi (amélioration de la convergence), soit un ensemble de solutions Pareto équivalent au point choisi (amélioration de la diversité).

La ***définition d'un sous-problème*** à partir d'un individu élu consiste à choisir d'une manière aléatoire les variables à passer au sous-problème et les autres variables sont fixées à leurs valeurs (*Voir exemple ci-dessous*). Après chaque recherche exacte on entame une phase de diversification maximale $\Delta \leftarrow \Delta_{max}$, une façon de réorienter la recherche vers un autre sous-espace éloigné. Notons que notre algorithme exact est autorisé à opérer sur un voisinage de taille limitée à 20. Ce choix est fait par expérimentation (au-delà de 20 le temps de calcul devient exponentiellement excessif).

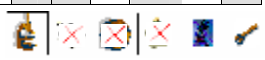
Exemple :

La définition d'un sous-problème peut être illustré par l'exemple suivant : n = 6 objets, k = 2 contraintes, p = 2 objectifs (voir la section 2 ci-dessus).



w^1	5	2	7	8	3	2	v^1	10	7	12	13	8	7	W^1	15	$Z^1(x)$	25	$\sum w^1 x$	10
w^2	3	8	9	11	5	6	v^2	10	8	6	14	13	7	W^2	22	$Z^2(x)$	30	$\sum w^2 x$	14
x	1	0	0	0	1	1													

La forme génotypique de la solution



La forme phénotypique de la solution

Figure 4.7 Illustration d'un environnement du problème global

On définit un voisinage d'ordre m = 4 autour de cette solution x, en choisissant aléatoirement $\frac{m}{2} = 2$ variables parmi les objets retenus (objets 1 et 6) et les $\frac{m}{2} = 2$ autres parmi les objets non retenus (objets 2 et 4), à passer au sous-problème. Les autres variables sont fixées à leurs valeurs (objets 3 et 5). Le sous-problème sera défini par un sous-ensemble des solutions réalisables qui diffèrent de x par la valeur d'une à m variables (un sous-ensemble du voisinage d'ordre m autour

de la solution *x*). La résolution exacte permet de trouver toutes les configurations de *xs* liée à la configuration *x/xs*, qui dominent (ou incomparable à) *x*.
vs : le vecteur valeur du sous problème
ws : vecteur poids du sous problème.
$\overline{v}$, $\overline{w}$ correspondent aux vecteurs *v* et *w* des objets fixés (objets 3 et 5).

Le sous-problème à résoudre par le B&B est défini comme suit (sachant que l'objet 6 est retenu) :

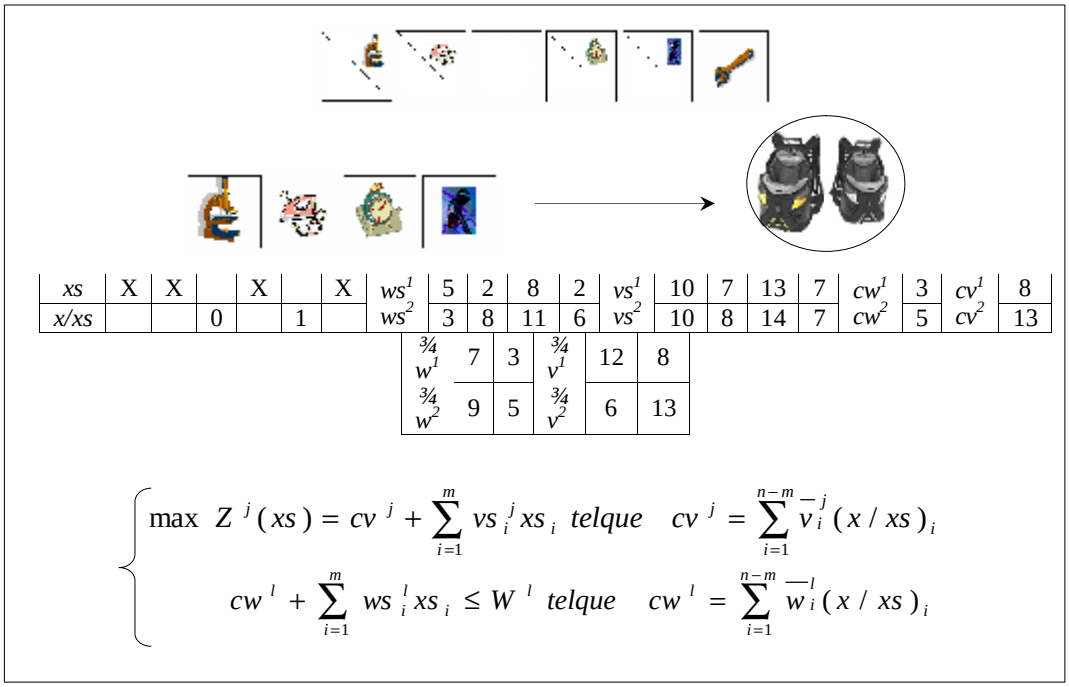


Figure 4.8 Illustration de la définition d'un sous problème à partir d'une solution ***x***

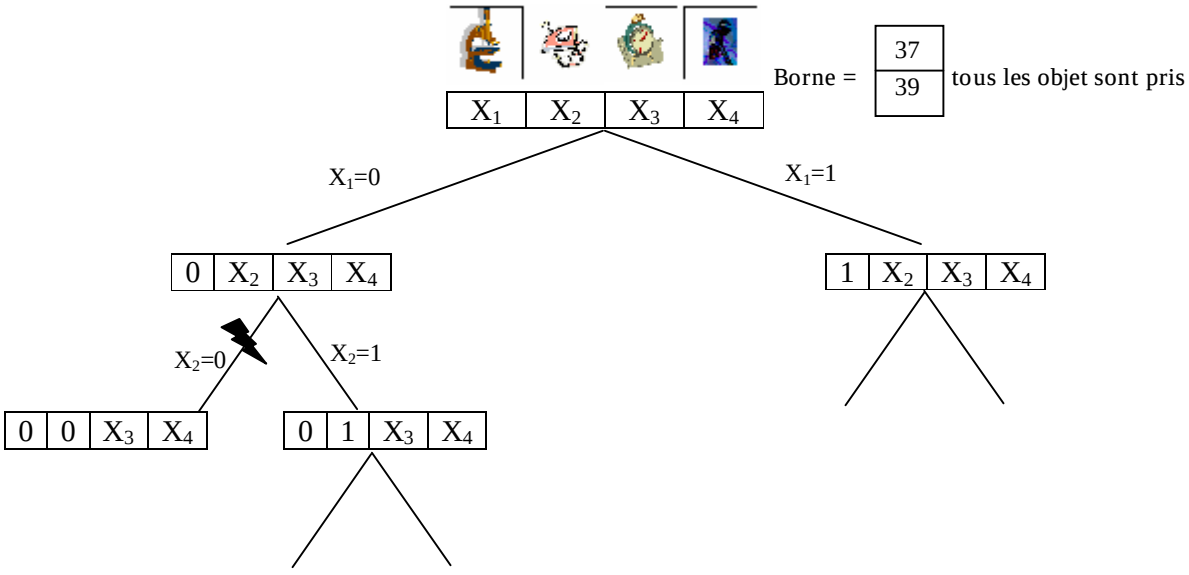


Figure 4.9 illustration de la construction de l'arbre de recherche

4.1 Stratégie et paramétrage

Lors de l'évaluation nous avons mis en place au préalable un service de test dédié à choisir les stratégies d'évolution (après avoir testé plusieurs) et à définir les valeurs des différents paramètres influant sur la qualité des résultats. Par exemple nous avons testé la méthode sans diversification (en fixant Δ à Δ_{min}) pour estimer le nombre de générations nécessaire à l'homogénéisation de la population, afin de régler le seuil de diversification. La stratégie du critère d'arrêt dont la population cesse d'évoluer pendant un certain nombre de générations successives permet de révéler la capacité de l'algorithme à poursuivre la recherche tout en améliorant les résultats. Enfin nous avons essayé de limiter la complexité de réglage des paramètres en corrélant les uns aux autres selon les résultats du test établi et des règles de bon sens.

- Taille de la population = 10, 20, 30 individus.
- $\Delta_{min} = 1$: La population n'accepte pas de clones.
- Δ_{max} est choisi en fonction de la taille du problème : ½, ¼ …
- Nombre de génération durant la diversification : nbr_gen_div=(%div* taille_pop)/(100*2), le pourcentage de la diversification est fixé à 60%, et 2 représente le nombre de descendants par génération.
- Seuil de diversification (nombre de génération sans amélioration nécessitant la variation de Δ) : fixé à la taille de la population.
- Seuil critique (nombre de génération sans amélioration critique nécessitant l'intervention du B&B) = nbr * Seuil_de_diversification, nbr = 2,3,...
- Critère d'arrêt : Plusieurs générations sans amélioration = nbr * seuil_critique telque nbr est un nombre entier.
- Stratégie de sélection : en mono-objectif, après avoir testé plusieurs stratégies (rang, roulette etc.) le tournoi à deux permet en générale de donner de meilleurs résultats. Dans le cas multiobjectif les parents sont choisis parmi la population externe avec une probabilité de ½, sinon parmi la population courante, et cela d'une manière aléatoire.
- Stratégie de remplacement : en phase d'intensification, l'individu généré remplace le plus mauvais individu dans la population. En multiobjectif, l'individu victime est choisi parmi le plus mauvais front (ranking). En phase de diversification, l'individu victime est choisi selon une méthode de tournoi à deux. En multiobjectif l'individu généré remplace dans la population l'individu qui lui est le plus semblable (crowding).
- Nombre de sous-problèmes généré = 10, 20,...
- Croisement à un point (aléatoire).

4.2 Résultats Expérimentaux

La méthode est programmée en Matlab 6.5, exécutée sur un PC sous Win XP (P4, 1.86 Ghz). Chaque instance de problème étant résolue dix fois par chaque algorithme. Nous avons gardé la meilleure résolution avec la moyenne du temps d'exécution.

Taille_pop	Seuil_diversificat	Seuil_critique
10	10	5*Seuil_diversificat
taille_voisinage_exact		Nbr_sous-prob-générés
10		10
Critère d'arrêt (la population cesse d'évoluer).		
- 100* Seuil critique = 5000 générations sans amélioration (**multidimensionnel**) - 10* Seuil critique = 500 générations sans amélioration (**multiobjectif)**		

Tableau 4.1. Les valeurs des différents paramètres utilisées dans l'expérimentation

4.2.1 Cas mono-objectif

Les expérimentations sont effectuées sur les mêmes instances que les travaux de K.H.Han et J.H.Kim (Han, 2000) obtenus par leur algorithme génétique quantique :

- Les poids w_i des objets sont aléatoirement générés entre [1, 10].
- Les valeurs v_i associés : $v_i = w_i + 5$.
- La moyenne de la capacité du sac à dos a été utilisée : $W = 1/2 \sum_{i=1}^{n} w_i$.
- fixé à 500 générations.

La table suivante montre les résultats expérimentaux des problèmes de sac à dos avec 100, 250, et 500 éléments. Nous avons comparé les résultats obtenus avec le meilleur algorithme des deux auteurs à savoir l'algorithme génétique quantique de 10 chromosomes :

Dimension	QGA (10)	MA|PM		(MA|PM)ÄB&B	
	val de Z	val de Z	Temps	val de Z	Temps
100	612,5	614,61	4,45s	**619,61**	7,73s
250	1480,3	1500,1	8,48s	**1510,2**	12,21s
500	2860,0	2910,8	14,29s	**2975,4**	18,95s

Tableau 4.2 *Résultats des différentes méthodes (mono-objectif).*

Les résultats obtenus par le (MA|PM)⊗B&B sont supérieurs en terme de qualité.

4.2.2 Cas multidimensionnel

les tests ont été effectués sur des problèmes d'un nouvel ensemble d'instances référencé par OR-library à l'adresse hces.bus.olemiss.edu/tools.html.

Dimension (n x k)	Best known solution	MA|PM		(MA|PM)⊗B&B	
		Val Z	Temps	Val Z	Temps
mkgk01 (100x15)	**3766**	3759	49s	**3766**	1m 3s
Mkgk02 (100x25)	**3958**	3946	1m 3s	**3958**	2m 44s
mkgk03 (150x25)	**5650**	5638	1m 47s	5646	3m 41s
mkgk04 (150x50)	**5764**	5748	2m 15s	5758	3m 56s
mkgk05 (200x25)	**7557**	7520	2m 49s	7545	4m 24s

Tableau 4.3 *Résultats des différentes méthodes (multidimensionnel).*

Les résultats obtenus par le (MA|PM)⊗B&B sont supérieurs en terme de qualité par rapport à MA|PM. Le critère d'arrêt dont la population cesse d'évoluer pendant 5000 générations permet de révéler la capacité de l'algorithme à poursuivre la recherche tout en améliorant le résultat.

4.2.3 Cas multiobjectif

Les expérimentations sont effectuées sur des instances du MOKP publiées par Zitzler à l'adresse http://www.tik.ee.ethz.ch/~zitzler/testdata.html. Nous avons choisi ici des instances bi-objectifs dont les résultats sont représentables graphiquement.

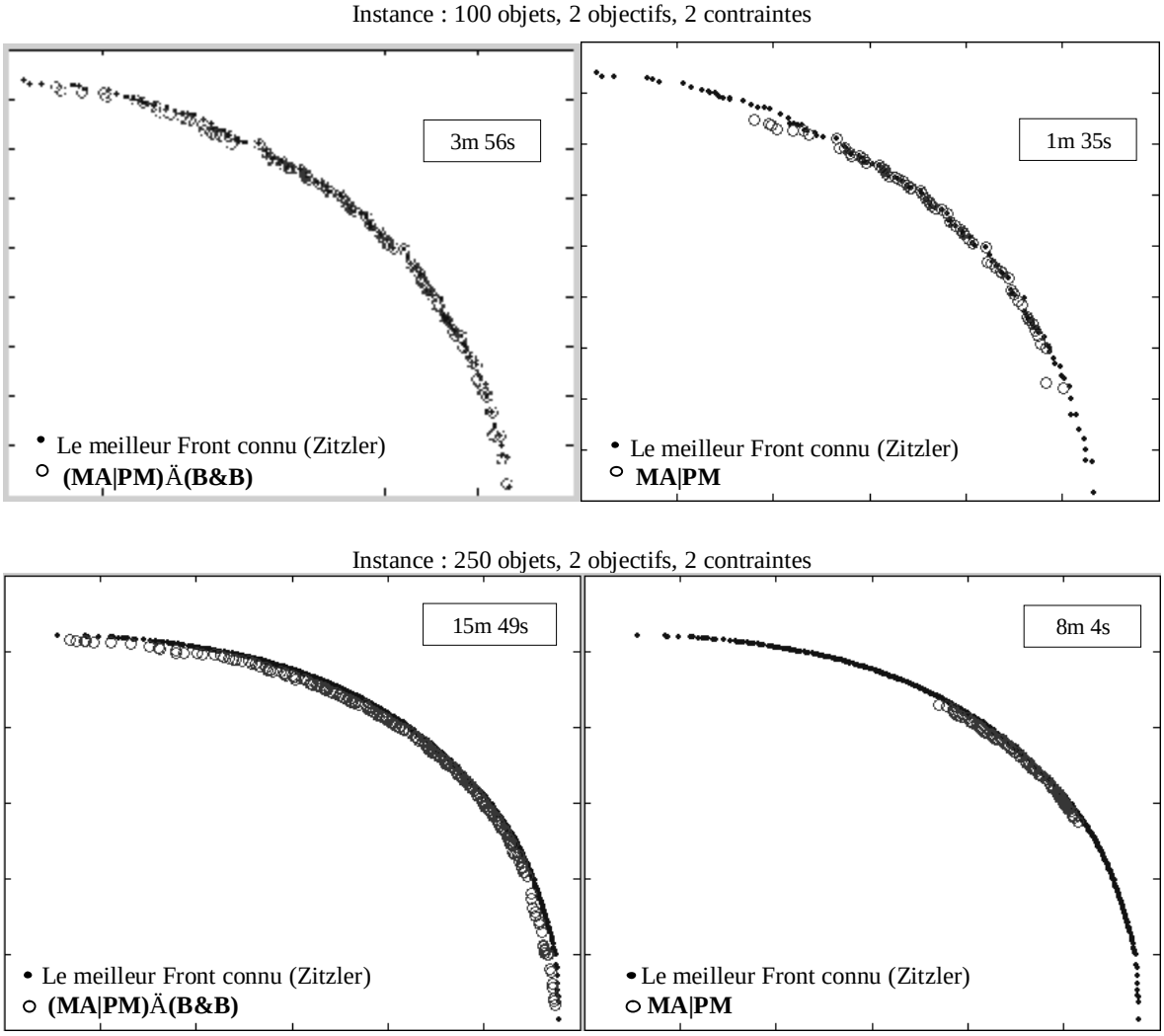


Figure 4.7 Résultats des différentes méthodes (SPEA, MA | PM et (MA | PM)Ä(B&B))

Nous observons que les deux algorithmes donnent de meilleurs résultats en terme de convergence, cela est dû principalement aux mécanismes de convergence utilisés (ranking et élitisme) et au très bon comportement de l'algorithme MA|PM et son pouvoir de poursuivre l'exploration dans d'autres zones de l'espace, permettant à la recherche de ne stagner (critère d'arrêt) qu'après une très bonne convergence. Notons aussi l'apport minime du crowding de base utilisé. Lors de nos expérimentations, Il nous est apparu que le choix des points extrêmes autour desquels on explore d'une manière exacte ses voisinages, se révèle un bon facteur de recherche (intensification et diversification) qui permet de 'creuser' et répartir les solutions le long de la

frontière Pareto. Nous avons appliqué le B&B sur un seul point choisi aléatoirement à partir du front Pareto, et les résultats obtenus sont inférieurs en diversification par rapport à ceux obtenus en utilisant les points extrêmes. Nous avons appliqué le B&B sur un seul point choisi aléatoirement à partir du front Pareto, et les résultats obtenus sont inférieurs en diversification par rapport à ceux obtenus en utilisant les points extrêmes (*figure* 4.8).

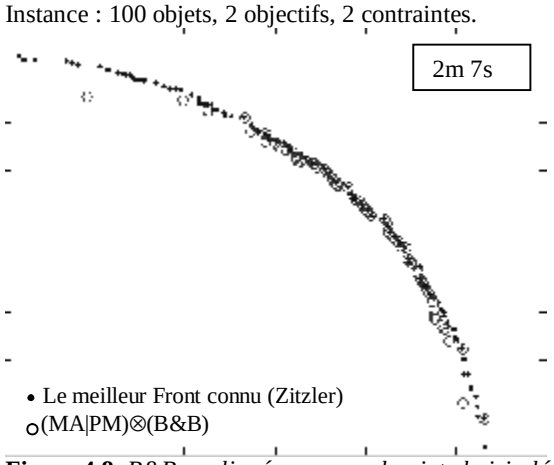


Figure 4.8 *B&B appliqué sur un seul point choisi aléatoirement sur le front Pareto (influence de la recherche en écartant le choix des points extrêmes)*

Le (MA|PM)⊗B&B est toujours meilleur en terme de qualité par rapport à MA|PM sur tous les problèmes testés. L'intervention du B&B permet soit d'améliorer la meilleure solution courante, sinon nous avons la même valeur obtenue par le MA|PM. En effet, l'ajout d'une recherche purement exacte à une métaheuristique selon notre schéma améliore les résultats obtenus par la métaheuristique (au pire des cas nous avons le même résultat mais avec un temps de calcul relativement élevé). Il est intéressant de noter les très bons résultats de (MA|PM)⊗(B&B) en terme de qualité, cependant, l'intervention du B&B multiplie par deux, le temps d'exécution. Ce temps de calcul étant en partie fonction du code (peut être optimisé) généré par l'interpréteur *Matlab*. Les algorithmes (SSGA, B&B, DD) utilisés sont implémentés de manière standard (de base). Pour cette raison, nous sommes convaincus que beaucoup d'améliorations sont possibles, non seulement par une meilleure programmation ou le choix d'un autre langage compilé, mais également, par l'utilisation des techniques avancées ou un meilleur réglage des divers paramètres de la méthode. Finalement, il est intéressant de noter que plusieurs aspects peuvent être étudiés en fonction des paramètres déterminants.

Conclusion

Afin de répondre à l'objectif de concevoir un algorithme d'optimisation combinatoire multiobjectif efficace, en temps de calculs (raisonnable), en qualité de solutions produites (l'intensification et la diversification) et permettant de traiter des problèmes de grande taille ; nous avons proposé une méthode coopérative méta/exacte Pareto, que nous avons appliquée au problème académique du sac à dos. Pour cela nous avons utilisé :

– Un algorithme de recherche globale SSGA pour bien explorer l'espace de recherche. Ce qui s'avère intéressant pour les problèmes de grandes tailles.

- Une recherche locale LS (la plus grande descente) appliquée à chaque nouvelle solution produite pour mieux exploiter les bonnes solutions trouvées.
- Une gestion contrôlée de la diversification de la population PM pour respecter l'équilibre souhaité entre l'intensification (l'amélioration des solutions) et la diversification (la propriété de poursuivre la recherche dans d'autres zones).
- Une recherche exacte de type B&B au pouvoir de recherche absolu, pour mieux exploiter le voisinage de la frontière Pareto.
- Des mécanismes de recherche : ranking, crowding et élitisme pour mieux converger et répartir les solutions le long de la frontière de Pareto trouvée.

Les résultats obtenus mettent en évidence le très bon comportement de notre méthode et montre l'efficacité de l'approche méta/exacte. En effet, l'ajout d'une recherche purement exacte à une métaheuristique selon notre schéma améliore les résultats obtenus par la métaheuristique (au pire des cas nous avons le même résultat mais avec un temps de calcul relativement élevé). Ce travail peut être amélioré notamment au travers d'application ou d'extension des propriétés avancées, comme l'utilisation d'un algorithme génétique adaptatif (pour réduire le problème de réglage des paramètres) ou la mise au point d'une méthode de simulation permettant l'analyse de l'approche en jouant sur les paramètres pour en extraire les meilleures valeurs.

Bibliographie

- [Applegate, 1998]
On the solution of the travelling salesman problem. D.Applegate, R.Bixby, V.chvàtal and W.Cook. Documenta Mathematica, Extra Volume ICM III :645-656, 1998.
- [Augerat, 1998]
separating capacity constraints in the CVRP using tabu search. P.Augerat, J.M.Belenguer, A.corber'n and D.Naddef. European Journal of Operational research, 106(2-3) :546-557, April 1998
- [Barichard, 2003]
Approches hybrides pour les problèmes multiobjectifs. V.Barichard, Thèse de Doctorat 2003, école doctorale d'Angers.
- [Barichard, 2004]
Résolution D'un Problème D'analyse De Sensibilité Par Un Algorithme D'optimisation Multiobjectif. V.Barichar & J.K.Hao 2004**,** LERIA Université d'Angers, 2 Bd Lavoisier F-49045 Angers CEDEX 1-France {Vincent.Barichard}{Jin-Kao.Hao}@univ-angers.fr

- [Basseur, 2002]
Design of Multi-objective Evolutionary Algorithm: Application to the Flow-shop Scheduling Problem. M. Basseur, F.Seynhaeve and EG.Talbi. In Congress of evolutionary computation. CEC'02, pages 1151-1156, Honolulu, Hawaii, USA, May 2002.
- [Basseur, 2005]
Conception D'algorithmes Coopératifs Pour L'optimisation Multiobjectif : Application Aux Problèmes D'ordonnancement De Type Flow-Shop». M. Basseur 2005, Université des sciences et technologies de Lille U.F.R. D'I.E.E.A. thèse pour obtenir le grade de Docteur de l'U.S.T.L.
- [Bent, 2004]
A two-stage hybride local search for the vehicle routing problem with time windows. R.Bent and P.V.Hentenrych. transportation science, 38(4) :515-530, Novembre 2004.
- [Berro, 2001]

Optimisation multiobjectif et stratégie d'évolution en environnement dynamique. Alin Berro 2001 thèse de doctorat.

– [Burke, 2001]
Effective local and guided variable neighborhood search methods for the asymmetric travelling salesman problem. E. K. Burke, P. I. Cowling, and R. Keuthen. In E. Boers et al., editors, Applications of Evolutionary Computing: EvoWorkshops 2001, volume 2037 of LNCS, pages 203–212. Como, Italy, April 2001. Springer-Verlag.

– [Carraway, 1990]
R.L. Carraway, T.L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. European Journal of Operational Research, 44 :95–104, 1990.

– [Charbier, 2002]
Coopération entre génération de colonnes sans cycle et recherche locale appliquée au routage de véhicules. A.Charbier, E.Danna and C.Le Pape. In journées nationales sur la résolution pratique de problèmes NP-complets (JNPC'2001), 2002

– [Chu, 1998]
A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics. P.C.Chu and J. E. Beasley. 4:63–86, 1998.

– [Coello, 1998]
An updated survey of G.A. based multiobjective optimization techniques. C.A.Coello Coello. Technical report, Lania-RD-98-08, Xalapa, Veracruz, Mexico, 1998.

– [Coello, 2002]
Evolutionary algorithms for solving multi-objective problems. C.A.Coello Coello, D.A.Van Veldhuizen, and G.B. Lomont, Kluwer Academic Press, 2002.

– [Coello, 2004]
Applications of Multi-Objective Evolutionary Algorithms. C.A.Coello Coello and G.B.Lamont. World Scietific Publishing co, 2004.

– [Corne, 2001]
PESA II: Region-based Selection in Evolutionary Multiobjective Optimization. D.W.Corne and al, in Procedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), p. 283-290, San Francisco, California, 2001.

– [Cotta, 1995]
Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP. C.Cotta, J.F.Aldana, A.J.Nebro and J.M.Troya. In D.W.Pearson, N.C.Steele, and R.F.Albrecht, editors, Artificial Neural Nets and Genetic Algorithms 2, page 277-280. Springer-Verlag, 1995.

– [Dahl, 1995]
A tabu search approach to the channel minimization problem. G.Dahl, K.Jornsten, and A.Lokketangen. In Liu, G., Phua, K.-H.,Ma, J., Xu, J., Gu, F., and He, C., editors, *Optimization - Techniques and Applications, ICOTA'95*, volume 1, pages 369–377, Chengdu, China. World Scientific.

– [Dhaenens, 2005]

Optimisation Combinatoire Multiobjectif: Apport Des Méthodes Coopératives Et Contribution A L'extraction De Connaissances. C.Dhaenens-Flipo. Université des sciences et technologies de Lille U.F.R. D'I.E.E.A. thèse pour obtenir le grade d'habilitation à diriger des recherches de l'U.S.T.L. 2005.

– [Deb, 2000]
A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multiobjective Optimization: NSGA II. K.Deb. Parallel problem solving form Nature – PPSN VI, springer lecture notes in computer science, p. 849-858, 2000.

– [Deb, 2001]
Controlled elitist non-dominated sorting genetic algorithms for better convergence. K. Deb and T. Goel. In Proceedings of Evolutionary Multi-Criterion Optimization. pages 67-81, 2001.

– [Dumitrescu, 2003]
Combinations of local search and exact algorithms. I.Dumitrescu & T.Stützle EvoWorkshops 2003, Volume 2611 of lecture notes in computer science, Essex, UK.

– [Dupas, 2004]
Amélioration de performance des systèmes de production : apport des algorithmes evolutionnistes aux problèmes d'ordonnancement cycliques et flexibles. Rémy Dupas 2004. Habilitation à diriger des recherches presentee a l'universite d'Artois.

– [Filho, 2000]
Constructive genetic algorithm and column generation: an application to graph coloring. G. R. Filho and L. A. N. Lorena. In Proceedings of APORS 2000 - The Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS, 2000.

– [Fischetti, 2003]
Local Branching. M. Fischetti and A. Lodi. *Mathematical Programming Series B*, 98:23–47, 2003.

– [Fogel, 1966]
Artificial Intelligence through Simulated Evolution. Fogel L.J., Owens A. J. and Walsh M. J., New York: Wiley, 1966.

– [Fonseca, 1995]
Multiobjective genetic algorithms with applications to control engineering problems. C.Fonseca 1995, PhD thesis, University of Sheffield.

– [Fourman, 1985]
Compaction of symbolic layout using genetic algorithm. M.P.Fourman. in procedings of the first international conference on genetic algorithms (ICGA)

– [French, 2001]
Using a hybrid geneticalgorithm/branch and bound approach to solve feasibility and optimization integer programming problems. A. P. French, A. C. Robinson, and J. M.Wilson. Journal of Heuristics, 7:551–564, 2001.

– [Friesz, 1993]

The multiobjective equilibrium network design problem revisited : A simulated annealing approach. T.Friesz, G.Anandalingam, N.Mehta, K.Nam, S.Shah, and R.Tobin. *EuropeanJournal of Operational Research*, 65 :44–57.

– [Fujimura, 1996]
Path planning with multiple objectives. K.Fujimura. *IEEE Robotics and Automation Society Magazine*, 3(1) :33–38.

– [Fujita, 1998]
Multi-objective optimal design of automotive engine using genetic algorithm. K.Fujita, N.Hirokawa, S.Akagi, S.Kimatura, and H.Yokohata, (1998). In *Design Engineering Technical Conferences DETC'98*, pages 1–11, Atlanta, Georgia.

– [Goldberg, 1987]
Genetic algorithms with sharing for multimodal function optimization. In Genetic Algorithms and their Applications: D.E. Goldberg and J. Richardson. Proceedings of the Second International Conference on Genetic Algorithms, pages 41- 49. Lawrence Erlbaum, 1987.

– [Goldberg, 1989]
Genetic Algorithms in Search, Optimisation and Machine Learning. Goldberg D.E. Addison Wesley publishing compagny, 1989.

– [Halhal, 1997]
Water network rehabilitation with a structured messy genetic algorithm. D.Halhal, G.Walters, D.Ouazar, and D.Savic. *Journal of Water Resources Planning and Management*, 123(3) :137–146.

– [Han, 2000]
Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem. Han.K.H, Kim.J.H, 2000, Dept. of Electrical Engineering, KAIST, 373-1, Kusong-dong Yusong-gu Taejon, 305-701, Republic of Korea. Khhan@vivaldi.kaist.ac.kr et johkim@vivaldi.kaist.ac.kr.

– [Holland, 1962]
Outline for a logical theory of adaptive systems. J.H.Holland. ACM Journal 9, pages 297-314, 1962.

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

– [Horn, 1993]
Multiobjective Optimization using the Niched Pareto Genetic Algorithm. J.Horn and N.Nafpliotis. Illigal TR, n° 93005, July 1993.

– [Ishibuchi, 1998]
A milti-objective genetic local search algorithm and its application to the flowshop scheduling. H.Ishibuchi and T.Murata, IEEE transactions on systems, Man and Cybernetics – Part C : applications and reviews, 28 :392-403, 1998.

– [Jahuira, 2002]

Hybride genetic algorithm with techniques applied to TSP. C.A.R.Jahuira. In second international workshop on intelligent systems design and application, pages 119-124. dynamic Publishers, 2002.

– [Jozefowiez, 2004]
Modélisation et résolution approchée de problèmes de tournées de véhicules. N.Jozefowiez 2004, PhD thesis, Université des Sciences et Technologies de Lille.

– [Klau, 2004]
Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. G.Klau, I.Ljubié, A.Moser, P.Mutzel, P.Neunez, U.Pferschy, G.Raidl and R.Weiskircher. In K.deb et al., editors, Genetic and Evolutionary Computation – GECCO 2004, volume 3102 of LNCS, page 1304-1315. Springer, 2004.

– [Klepeis, 2003]
Hybrid global optimization algorithms for protein structure prediction: Alternating hybrids. J.L.Klepeis, M.J.Pieja and C.A.Floudas. Biophysical Journal, 4(84) :869-882, February 2003.

– [Knowless, 1999]
The Pareto Archived Evolution Strategy: A new Baseline Algorithm For Multiobjective Optimisation. J.D.Knowles and D.W.Corne. Congress on Evolutionary Computation, p. 98-105, Washington, July 1999.

– [Knowless, 2000]
Pareto-Envelope based Selection Algorithm for Multiobjective Optimization. J.D.Knowles, D.W.Corne and M.J.Oates. In proceedings of the sixth international conference on parallel problem solving from nature (PPSN VI), p. 839-848, Berlin, septembre 2000.

– [Kostikas, 2004]
Genetic programing applied to mixed integer programming. K.Kostikas and C.Fragakis. In euroGP 2004, volume 3003 of lecture Notes in Computer Science, pages 113-124, 2004

– [Koza, 1992]
Genetic Programming. Koza J.R. Cambridge MA, MIT Press, 1992.

– [Loughlin, 1997]
The neighborhood contraint method : A genetic algorithmbased multiobjective optimization technique. D.Loughlin, S.Ranjithan. In Back, T., editor, *Seventh Int. Conf. on Genetic*

– [Maniezzo, 1999]
Exact and approximate nondeterministic tree-search procedures for quadratic assignment problem. INFORM Journal on Computing, 11(4) :358-369, 1999.

– [Meunier, 2000]
A multiobjective genetic algorithm for radio network optimization.H.Meunier, EG.Talbi, P.Reininger. In CEC, volume 1, page 317-324. Piscataway. New jersey. July 2000. IEEE service center.

– [Meunier, 2002]
Algorithmes évolutionnaires parallèles pour l'optimisation multi-objectif de réseaux de télécommunications mobiles. Hervé Meunier 2002 Thèse de doctorat.

– [Michalewicz, 1996]
Genetic Algorithms + Data Structures = Evolution Programs. Z. Michalewicz. 3rd edn, Springer Verlag, Berlin Heidelberg New York, 1996.

– [Morse, 1980]
Reducing the Size of Nondominated Set : Pruning by Clustering. J.n.morse. Computers and operation research. 7(1-2), p. 55-66. 1980.

– [Obayashi, 1998]
Niching and elitist models for multiobjective genetic algorithms. S.Obayashi, S.Takahashi, and Y.Takeguchi, (1998). In *Parallel Problem Solving from Nature PPSN'5*, pages 260–269,Amsterdam. Springer-Verkag.

– [Padberg, 1991]
A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. M. Padberg and G.Rinaldi. SIAM review, 33(1) :60-100, 1991.

– [Palpant, 2001]
Conception d'une métaheuristique et application au problème d'ordonnancement de projet à moyens limités. M.Palpant sous la direction de C.Artigues et P.Michelon. Laboratoire d'Informatique d'Avignon 2001.

– [Portmann, 1998]
Branch and bound crossed with GA to solve hybride flowshops. M.C.Portmann, A.Vignier, D.Dardihac and D.Dezalay. European Journal of Operational Research, 107(2) : 389-400, 1998.

– [Puchinger, 2004]
Models and algorithms for three-stage two-dimensional bin packing. J. Puchinger and G. R. Raidl. Technical Report TR 186–1–04–04, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2004. submitted to the European Journal of Operations Research.

– [Puchinger, 2005]
Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. Jakob Puchinger and Günther R. Raidl 2005. Institute of Computer Graphics and Algorithms Vienna University of Technology, Vienna, Austria {puchinger|raidl}@ads.tuwien.ac.at

– [Raidl, 1998]
An improved genetic algorithm for the multiconstrained 0–1 knapsack problem. G. R. Raidl. In D. B. Fogel, editor, Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, pages 207–211. IEEE Press, 1998.

– [Schaffer, 1984]
Multiple objective optimization with vector evaluated genetic algorithms. J.D. Schaffer. PhD thesis, Vanderbilt University, 1984.

– [Schott, 1995]
Fault tolerant design using single and multicriteria genetic algorithm optimization. J.R. Schott.Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1995.

– [Schweffel, 1995]
Evolution and Optimum Seeking. Schweffel H-P, New York, Wiley 1995.

– [Sen, 1988]
A branch and bound approach to the bicriterion scheduling problem involving total flowtime and range of lateness. Sen, T., Raiszadeh, M., and Dileepan, P. 1988. *Management Science,* 34(2) :254–260.

– [Sevaux, 2004]
Métaheuristiques Stratégies Pour L'optimisation De La Production De Biens Et De Services. M.Sevaux 2004, Thèse de Doctorat.

– [Shaw, 1996]
Initial study of multi-objective genetic algorithms for scheduling the production of chilled ready meals. K.Shaw, P.Fleming. In *Mendel'96 2nd Int. Conf. on Genetic Algorithms,* Brno, Czech Republic.

– [Shaw, 1998]
Using constraint programming and local search methods to solve vehicle routing problems. In principales and practice of constraint programming- CP98, 4th International Conference, volume 1520 of lecture notes in computer science, pages 417-431, Pisa, Italy, 1998. Springer-Verlag.

– [Staggemeier, 2002]
A hybrid genetic algorithm to solve a lot-sizing and scheduling problem. A. T. Staggemeier, A. R. Clark, U. Aickelin, and J. Smith. In Proceedings of the 16th triannual Conference of the International Federation of Operational Research Societies, Edinburgh, U.K., 2002.

– [Stewart, 1991]
B.S. Stewart and C.C. White. Multiobjective A*. Journal of the ACM, 38(4) :775–814, 1991.

– [Talbi, 2000]
Une taxinomie des métaheuristiques hybrides. E.G.Talbi, ROADEF 2000.

– [T'kindt, 2002]
An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. V.T'kidt, N.Monmarché, F.Tercinet and D.Laügt. European Journal of Operational Research, 142 :250-257, 2002.

– [Todd, 1997]
A multiple criteria genetic algorithm for container loading. D.Todd, P.Sen.In Back, T., editor, *Seventh Int. Conf. on Genetic Algorithms ICGA'97,* pages 674–681, San Mateo, California. Morgan Kaufmann.

– [Ulungu, 1995]
The two phases method : An efficient procedure to solve biobjective combinatorial optimization problems. E.L. Ulungu and J. Teghem. Foundation of computing and decision science, 20 :149–156, 1995.

– [Veldhuizen, 1997]
Finding improved wire-antenna geometries with genetic algorithms. D.V.Veldhuizen, B.Sandlin, R.Marmelstein, G.Lamont, and A.Terzuoli. In Chawdhry, P., Roy, R., and Pant, P., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240,London. Springer Verlag.

– [Veldhuizen, 2000]
On measuring multi-objective evolutionary algorithm performance. D.A.V Veldhuizen and G.B.Lamont. In 2000 Congress of evolutionary computation. Piscataway. New jersey. Volume 1, page 204-211, July 2000.

– [Weinberg, 2001]
A co-evolutionary meta-heuristic for the assignment frequencies in cellular networks. B.Weinberg. In *EvoCop*.

– [Woodrurff, 1999]
A chunking based selection strategy for integrating meta-heuristics with branch and bound. D. L. Woodruff. In S. Voss et al., editors, Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization, pages 499–511. Kluwer Academic Publishers, 1999.

– [Zitzler, 1999]
Evolutionary algorithms for multiobjective optimization: methods and applications. E. Zitzler. PhD thesis, Swiss Federal Institute of Technology (Zurich), 1999.

– [Zitzler, 1998]
An evolutionary algorithm for multiobjective optimization: the stength Pareto approach. Technical report, Swiss Federal Institute of Technology E. Zitzler and L. Thiele. (Zurich), 1998.