CHAPITRE 7

UNE APPROCHE HYBRIDE POUR L'OPTIMISATION MULTI-OBJECTIFS : GENETIC IMMUNE STRATEGY FOR MULTIPLE OBJECTIVE OPTIMIZATION (GISMOO)

7.1 Introduction

Dans la pratique, les situations industrielles rencontrées correspondent rarement à des problèmes nécessitant l'optimisation d'un seul objectif. En effet, la plupart des problèmes d'optimisation combinatoire rencontrés dans l'industrie nécessitent l'optimisation simultanée de plusieurs objectifs souvent conflictuels. Le problème industriel d'ordonnancement de voitures traité dans cette thèse illustre bien cet état de fait. Toutefois, malgré l'intérêt indéniable d'aborder les problèmes industriels d'un point de vue multi-objectifs, plusieurs auteurs ont noté [McKay et Wiers 1999; Yang et Liao 1999], en recensant la littérature, que les chercheurs s'attardaient principalement à des contextes théoriques de base.

Dans ce chapitre ainsi que dans le suivant, nous allons nous intéresser à la résolution de problème d'optimisation combinatoire multi-objectifs au sens Pareto sans aggrégation des objectifs contrairement à l'approche lexicographique présentée au chapitre précédent. En particulier, nous proposons GISMOO, un nouvel algorithme Pareto générique pour traiter les problèmes d'optimisation multi-objectifs. Cet algorithme combine des concepts issus des algorithmes génétiques avec des paradigmes issus des systèmes immunitaires artificiels. L'originalité de cette approche réside, d'une part, dans la manière dont les concepts issus des deux domaines sont combinés et utilisés dans un même algorithme. D'autre part, Coello Coello et Cortés [2005] ont noté que jusqu'à récemment les efforts pour étendre les approches basées sur des systèmes immunitaires à l'optimisation combinatoire multi-objectifs étaient pratiquement inexistants. Dans cette optique, GISMOO représente donc une contribution visant à combler en partie cette lacune. GISMOO est donc un nouvel

algorithme multi-objectifs au sens Pareto. Afin d'évaluer ses performances, nous devons le comparer avec d'autres algorithmes multi-objectifs de la littérature. Le problème industriel d'ordonnancement de voitures présenté au Chapitre 2 est un problème où trois objectifs conflictuels sont à minimiser. Toutefois, à notre connaissance, ce problème a toujours été traité dans la littérature en considérant les trois objectifs de manière lexicographique. Les principaux algorithmes multi-objectifs n'ont donc, dans leur version intégrale, jamais été appliqués à la résolution du POVI. C'est pourquoi nous avons choisi, dans un premier temps, de valider notre algorithme en analysant ses performances à l'aide d'un problème classique largement étudié dans la communauté multi-objectifs, le problème de sac à dos multidimensionnel multi-objectifs [Zitzler et Thiele 1998; Jaszkiewicz 2000; Knowles et Corne 2000; Zitzler *et al.* 2001; Barichard 2003; Zinflou *et al.* 2006]. Dans le chapitre suivant, nous analysons, dans un deuxième temps, les performances de GISMOO sur le POVI. Ainsi, nous validons les performances de notre approche aussi bien sur des problèmes multi-objectifs théoriques que sur des problèmes multi-objectifs réels. De plus, il est important de préciser que les tests effectués dans ces deux chapitres sont réalisés en suivant les standards adoptés par la communauté multi-objectifs [Coello Coello *et al.* 2002].

Le reste du présent chapitre est organisé de la manière suivante. On passe brièvement en revue à la Section 7.2 les principales méthodes basées sur les SIA pour l'optimisation multi-objectifs. La Section 7.3 introduit GISMOO un nouvel algorithme Pareto qui combine plusieurs concepts issus des algorithmes génétiques et des systèmes immunitaires artificiels d'une manière unique. Par la suite, nous présentons brièvement, à la Section 7.4,

le problème d'application qui est utilisé dans ce chapitre : *le problème de sac à dos multi-objectifs*. Finalement, la Section 7.5 présente les résultats expérimentaux de l'approche proposée, dans le contexte du problème multi-objectifs de sac à dos multidimensionnel, en les comparant à ceux d'autres algorithmes de la littérature.

7.2 Résolution de problèmes multi-objectifs à l'aide de SIA

Dans le passé, les systèmes immunitaires artificiels (SIA), que l'on a présenté à la Section 3.4 du Chapitre 3, ont démontré leur potentiel pour maintenir la diversité dans la population d'un algorithme génétique pour l'optimisation multimodale [Forrest et Perelson 1991; Smith *et al.* 1992; Smith *et al.* 1993]. Smith *et al.* [1993] montrent, dans leurs travaux qu'une forme de partage de la performance émerge lorsqu'on combine un algorithme génétique avec un système immunitaire artificiel. Les SIA ont aussi été combinés à des algorithmes évolutionnaires pour résoudre des problèmes de satisfaction de contraintes [Cui *et al.* 2001].

Toutefois, on note que très peu d'auteurs ont proposé des méthodes basées sur des systèmes immunitaires artificiels pour résoudre des problèmes d'optimisation combinatoire multi-objectifs [Coello Coello et Cortés 2005]. À notre connaissance, une des premières approches basées sur les SIA pour résoudre des problèmes d'optimisation multi-objectifs a été proposée par Yoo et Hajela [1999]. Cette approche hybride un algorithme génétique avec un SIA. Dans leur algorithme, les auteurs utilisent une fonction d'agrégation linéaire pour combiner les objectifs et déterminer la performance des solutions générées par l'algorithme. Par la suite, les meilleures solutions sont désignées comme étant des antigènes

et le reste de la population constitue les anticorps du SIA. Finalement, une simulation d'une réponse immune est effectuée pour identifier les anticorps qui sont les plus proches des antigènes sélectionnés en terme de distance de Hamming. Les auteurs ont appliqué leur algorithme à la résolution de problèmes d'optimisation structurelle. Toutefois, les auteurs n'ont comparé leur algorithme à aucune autre technique de résolution dans leur étude.

Il est aussi important de mentionner l'algorithme CLONALG introduit par De Castro et Von Zuben [2002] qui est un SIA basé sur le principe de sélection par clonage [De Castro et Timmis 2002]. L'algorithme CLONALG a été développé pour résoudre des problèmes d'optimisation multimodale ou uni-objectif. De leur côté, Coello Coello et Cortés [2005] ont proposé le Multi-objective Immune System Algorithm (MISA) qui est considéré comme la première tentative réelle de résolution de problèmes d'optimisation combinatoire multi-objectifs générique à l'aide d'un SIA. Contrairement à l'approche de Yoo et Hajela [1999], MISA n'est pas un algorithme hybride et, comme CLONALG, il se base sur le principe de sélection par clonage. Plus récemment, Tavakkoli-Moghaddam *et al.* [2007] proposèrent une autre approche immune pour résoudre un problème de flow-shop bi-objectifs. Comme CLONALG et MISA, l'approche de Tavakkoli-Moghaddam et al. se base aussi sur le principe de sélection par clonage. Les résultats obtenus par ces trois algorithmes suggèrent que le principe de sélection par clonage est une voie prometteuse en optimisation multi-objectifs où beaucoup de travail reste à faire.

7.3 Genetic Immune Strategy for Multiple Objectives Optimization (GISMOO)

Dans cette section, nous présentons GISMOO, un nouvel algorithme combinant un algorithme génétique élitiste utilisant des concepts de dominance Pareto avec les concepts de sélection par clonage et d'hyper mutation issus des systèmes immunitaires artificiels. L'originalité de cette approche réside principalement dans la manière avec laquelle la métaphore immune est utilisée, dans un algorithme génétique Pareto, pour identifier et mettre l'emphase sur les régions de l'espace de solution peu exploitées lors des itérations de l'algorithme. L'ajout d'une phase immune favorise ainsi une plus grande diversité au sein de la population de notre algorithme multi-objectifs. De plus, même si l'hybridation entre AG et SIA n'est pas en soit une nouveauté, GISMOO représente cependant, à notre connaissance, un des premiers algorithmes Pareto générique hybridant algorithmes génétiques et systèmes immunitaires artificiels dans un même algorithme pour résoudre des problèmes d'optimisation combinatoire multi-objectifs.

7.3.1 Création de la population initiale

Dans l'approche proposée, les individus de la population initiale sont générés de deux façons : à 70 % de manière aléatoire et à 30 % en utilisant une heuristique gourmande liée au problème à résoudre.

7.3.2 Gestion de l'élitisme

GISMOO est un algorithme élitiste c'est-à-dire qu'il possède des mécanismes assurant la conservation des individus non dominés au cours de la recherche. Pour cela, GISSMO utilise le principe d'archive pour conserver les solutions non dominées rencontrées durant

la recherche. Toutefois, dans notre approche, tous les individus de l'archive ne participent pas au processus de sélection. En effet, dans l'algorithme GISSMO, seuls les individus non dominés encore présents dans la population courante participent au processus de sélection. Il est important de noter ici que la taille de l'archive de notre algorithme n'est pas fixe et n'est pas limitée à un nombre maximum d'individus. Finalement, pour s'assurer que les meilleurs individus de la population sont conservés dans la population courante, GISSMO utilise une procédure de remplacement déterministe décrite à la Section 7.3.6.

7.3.3 Assignation de la performance

Un des principes élémentaires de tous algorithmes multi-objectifs est de distribuer la population de manière homogène le long de la frontière Pareto. Dans ce but, l'assignation de la performance d'un individu est un élément fondamental pour le succès d'une approche. GISMOO est un algorithme qui exploite la notion de dominance au sens Pareto proposée par Goldberg [1989]. L'évaluation de la performance des individus se fait donc selon les concepts de *dominance* et d'*isolement*.

7.3.3.1 Le facteur de dominance

À chaque itération t, la première étape du calcul de la performance consiste à assigner à chaque individu x, de la population Parent (POP_t) combinée à la population d'enfants et de clones (Q_t), une force S(x) correspondant au nombre de solutions y dominées par x, tel que :

$$S(x) = \left| \{ y \mid y \in POP_t \cup Q_t, x \succ y \} \right| \tag{7.1}$$

où | | représente la cardinalité de l'ensemble et le symbole $\succ$ indique une relation de dominance de x par rapport à y. À partir de $S(x)$, le *facteur de dominance* d'un individu x, noté $R^+(x)$, est déterminé dans GISMOO à l'aide de l'équation suivante:

$$R^+(x) = \begin{cases} \dfrac{S(x)}{1+2*S(x)} & si \sum_{y \in POP_t \cup Q_t, y \succ x} S(y) = 0 \\ \sum_{y \in POP_t \cup Q_t, y \succ x} S(y) & sinon \end{cases} \tag{7.2}$$

Ainsi, les individus non dominés y pour lesquels $\sum_{y \in POP_t \cup Q_t, y \succ x} S(y) = 0$ n'ont pas tous un facteur de dominance identique et égal à 0, mais un facteur de dominance compris entre 0 et 0.5 en fonction du nombre de solutions qu'elles dominent. Pour les solutions non dominées, cette méthode d'assignation du facteur de dominance permet de mieux tenir compte de la distribution des solutions dominées des populations *POP* et *Q* dans l'espace de recherche. De cette manière, le calcul du facteur de dominance favorise les solutions non dominées situées dans des régions sous-exploitées.

On note des similitudes entre les mécanismes d'assignation de la performance du PMS^{MO} [Zinflou *et al.* 2006] et ceux utilisés par GISMOO. Toutefois, les populations considérées dans les deux algorithmes pour l'assignation de la performance ne sont pas les mêmes. En effet, dans GISMOO le calcul de la performance tient compte des populations Parent et Enfant courantes, alors qu'au niveau du PMS^{MO} le calcul se fait en considérant la population Parent courante et l'archive.

7.3.3.2 Le facteur d'isolement

Le calcul du facteur de dominance tel que présenté à la sous-section précédente est une technique de nichage basée sur la notion de dominance au sens Pareto. Toutefois, lorsque la plupart des individus évalués sont des solutions non dominées, ou ont les mêmes performances brutes, cette technique peut s'avérer inefficace. Afin d'éviter ce genre de situation et d'introduire un peu plus de diversité dans la population, une information additionnelle sur la densité de solutions entourant un individu x est calculée dans notre approche. Cette information supplémentaire correspond au *facteur d'isolement* (*Dist*(x)) de l'algorithme. Ce calcul se fait selon l'équation suivante :

$$Dist(x) = \min_y \left[\sqrt{(f_1(x) - f_1(y))^2 + \ldots + (f_{noj}(x) - f_{nobj}(y))^2} \right] \text{ avec } x \neq y \qquad (7.3)$$

où *nobj* indique le nombre d'objectifs du problème. On remarque que le calcul de la densité (*Dist*) de GISMOO s'inspire du calcul de la métrique d'espacement ***Sp*** [Schott 1995] et a pour objectif d'évaluer la distance séparant un individu x de son plus proche voisin. Il est important de mentionner que le calcul du facteur d'isolement se fait en considérant les individus des populations Parent et Enfant uniquement. Notons aussi que cette information additionnelle n'est pas directement incorporée au calcul de la performance d'un individu. En effet, le facteur d'isolement est utilisé, dans la phase génétique, pour différencier deux individus lors du processus de sélection en cas d'égalité sur le facteur de dominance et lors des procédures de tris des populations. Dans la phase immune, cette information additionnelle est aussi utilisée pour déterminer le nombre de clones à produire pour un individu x donné.

7.3.4 Sélection

Plusieurs schémas de sélection auraient pu être envisagés dans l'approche proposée dans ce chapitre. Toutefois, comme elle est peu coûteuse à mettre en œuvre et à exécuter et qu'elle a fait ses preuves avec les autres algorithmes génétiques proposés dans cette thèse, la procédure de sélection retenue dans la phase génétique de GISMOO est une sélection par tournoi binaire. Précisons toutefois que le tournoi est effectué en considérant le facteur de dominance des deux individus participants au tournoi et en cas d'égalité, l'égalité est brisée en utilisant le facteur d'isolement.

7.3.5 Phase immune

Dans l'approche proposée, la population Enfant (Q) est subdivisée en deux portions de même taille. Une sous-population d'enfants de taille $N/2$ générée en utilisant les opérateurs de sélection, croisement et mutation décrits précédemment et une sous-population de clones, elle aussi de taille $N/2$, générée selon le principe de sélection par clonage introduit par De Castro et Timmis [2002]. Le principe de sélection par clonage permet de modéliser le fait que seuls les meilleurs « *anticorps* » vont proliférer dans la population. Dans notre algorithme, les anticorps correspondent aux individus non dominés de la population courante. Ainsi, pour créer la sous-population de clones, la première étape consiste à trier la population Parent courante (POP_{t+1}) en front selon le même principe que l'algorithme du NSGAII [Deb 2000]. Les individus non dominés de POP_{t+1}, ceux situés dans le premier front, sont alors sélectionnés comme population d'anticorps à cloner. Dans GISMOO, le nombre de clones produit pour chaque anticorps n'est pas constant, mais est proportionnel à son degré d'isolement. Ainsi, plus un individu est éloigné des autres en terme de distance,

plus le nombre de clones qu'il génère est important. Par cette technique, on cherche à identifier et à mettre l'emphase sur les solutions non dominées situées dans des régions isolées. Le calcul du nombre de clones pour chaque anticorps x se fait comme suit :

$$nb\_clones(x) = round\left[\left(Dist(x)*\frac{N}{2}\right)\Big/\sum_{x=1}^{|Front_1|}Dist(x)\right] \tag{7.4}$$

Dans cette équation, *nb_clones(x)* indique le nombre de clones pour l'individu non dominé x, $|Front_1|$ donne le nombre d'individus non dominés de la population et $Dist(x)$ correspond au facteur d'isolement de l'individu x tel que décrit à la Section 7.3.3.2. La fonction *round* arrondit le résultat obtenu à l'entier le plus proche.

Une fois le nombre de clones à retenir pour un individu x déterminé, on produit à chaque itération de la phase d'expansion par clonage deux clones (c_1^{clo}, c_2^{clo}) qui sont des copies de l'individu x. Ces clones sont, par la suite, mutés en utilisant deux opérateurs de mutation différents adaptés au problème à résoudre afin d'obtenir (c_1^{mut}, c_2^{mut}). Après évaluation, les deux clones mutés sont comparés et on conserve le meilleur des deux (celui qui domine l'autre au sens Pareto). En cas d'égalité, on sélectionne un des deux de manière aléatoire. Le clone muté sélectionné est par la suite ajouté à la population Q courante. Ce processus est ainsi répété pour chaque individu x du 1er front jusqu'à ce que le nombre de clones sélectionnés pour x soit égal à nb_clones(x).

Comme dans l'algorithme MISA [Coello Coello et Cortés 2005], GISMOO utilise le principe de sélection par clonage dans sa phase immune. Toutefois, il existe des différences majeures entre les deux algorithmes notamment en ce qui concerne le nombre de gènes à muter, les taux de mutation, le nombre de clones à générer pour chaque individu ainsi que

les stratégies de sélection des individus à cloner. Par exemple, dans MISA le nombre de gènes à muter pour chaque clone varie en fonction du rang du clone sélectionné alors que, dans GISMOO, ce nombre est constant et aucune procédure de classement supplémentaire n'est nécessaire. L'algorithme MISA utilise un taux de mutation variable pour les « moins bons » anticorps ce qui n'est pas le cas de GISMOO. Dans notre approche, le nombre total de clones à produire est fixé à 50% de la taille de la population alors que, dans MISA, ce nombre est fixé à 600% de la taille de la population. Une autre différence majeure entre les deux algorithmes réside dans le fait que seuls les individus non dominés de la population courante sont sélectionnés pour être clonés dans GISMOO alors que, dans MISA, si le nombre de solutions non dominées est inférieur à 5% de la taille de la population, des individus dominés sont sélectionnés pour être clonés. On note aussi que, dans l'approche proposée dans ce chapitre, le nombre de clones pour chaque anticorps est proportionnel à son facteur d'isolement et est calculé à chaque itération selon l'Équation 7.4. À l'opposé, dans MISA, chaque anticorps produit le même nombre de clones au début de l'algorithme. Par la suite, le nombre de clones produits par chaque anticorps augmente ou diminue selon certaines règles en fonction de la taille de la population secondaire de l'algorithme [Coello Coello et Cortés 2005]. On constate aussi que les deux algorithmes utilisent des stratégies différentes de sélection des anticorps et d'ajout des clones à la population. Finalement, une autre différence de taille entre les deux approches est que GISMOO est une approche hybride alors que MISA est un SIA simple.

7.3.6 Remplacement

GISMOO est un algorithme élitiste. Dans cette optique et afin de conserver les meilleurs individus dans la population courante, la stratégie de remplacement utilisée est un remplacement déterministe de type $(\lambda+\mu)$ où λ indique la taille de la population Parent et μ la taille de la population Enfant. Dans ce schéma de remplacement, les populations Parent et Enfant sont jointes et triées et on ne conserve que les λ meilleurs individus afin de former la prochaine génération. Dans l'approche proposée, on a $\lambda=\mu=N$.

7.3.7 Algorithme général

Le pseudo-code présenté à l'Algorithme 7.1 décrit le fonctionnement général de l'approche GISMOO. Après l'initialisation des différentes variables, l'algorithme débute par la génération de la population initiale POP_0 (ligne 2). Chaque individu de cette population est évalué et une performance lui est assignée conformément aux mécanismes d'assignation de la performance décrits à la Section 7.3.3. Une fois la performance assignée, on génère une population initiale d'enfants Q_0 que l'on évalue. Le processus itératif de l'algorithme (lignes 7-41) commence par la phase génétique (lignes 8-24) en combinant les populations Parent et Enfant courantes. Les individus de la nouvelle population combinée sont évalués et leurs facteurs d'isolement calculé (lignes 9-11). À cette étape, on note que, pour des fins de normalisation, on calcule une approximation du *point idéal* et du *point Nadir* à partir des individus de l'archive (ligne 10). Les coordonnées du point idéal correspondent aux meilleures valeurs de chaque objectif des solutions de la frontière Pareto. À l'opposé, les coordonnées du point nadir correspondent aux pires valeurs de chaque objectif des solutions de la frontière Pareto. La population combinée est

ensuite triée et seuls les *N* premiers individus sont conservés pour former la prochaine génération courante. On retrouve, par la suite, les étapes classiques de sélection par tournoi, de croisement et de mutation d'un algorithme génétique classique. Notons ici que lors d'un croisement deux enfants peuvent être générés; dans ce cas, on compare les deux enfants l'un par rapport à l'autre à l'aide de la procédure *Compet* (ligne 22). L'algorithme 7.2 décrit la procédure *Compet* utilisée à la ligne 22 pour comparer deux solutions. Cette procédure reçoit en paramètres deux solutions *x* et *y* et retourne la meilleure des deux. Le meilleur des deux enfants est alors ajouté à la population Enfant (ligne 23). Ce processus est ainsi répété jusqu'à ce que *N/2* enfants soit ajoutés à la population en cours de création *Q*. Une fois la phase génétique terminée, la phase immune de GISMOO (lignes 25-39) débute par le tri de la population *POP* courante en fronts selon le même principe que le NSGAII. On calcule, par la suite, la somme des facteurs d'isolement des individus situés dans le premier front et on commence la phase d'expansion par clonage tel qu'indiquée à la Section 7.3.5 de manière à ajouter *N/2* clones à la population d'enfants. Il est important de mentionner que, à chaque itération de l'algorithme, chaque nouvel individu non dominé trouvé vient s'ajouter à l'archive et les individus de l'archive dominés par le nouvel arrivant sont supprimés. Ainsi, à la fin de l'algorithme, les individus contenus dans l'archive sont retournés au décideur (ligne 42).

Algorithme 7.1 : pseudo-code détaillé de GISMOO

1: Initialiser l'archive A à ∅ et t à 0
2: Générer la population initiale POP_t aléatoirement ou avec des heuristiques gourmandes
3: Évaluer chaque individu $x \in POP_t$et mettre à jour A
4: Trier POP_t
5: Générer une population initiale Q_t d'enfants de même taille que la population Parent par croisement
6 : Évaluer chaque nouvel enfant créé et mettre à jour A
7: **Tant qu'**aucun critère d'arrêt n'est atteint
8 : Joindre Q_t *et* POP_t
9 : Calculer de la performance de chaque individu de la population combinée
10 : Évaluer le point idéal courant et le point Nadir courant à partir de A
11 : Calculer les distances normalisées entre chaque solution
12 : Trier $Q_t \cup POP_t$
13 : Conserver les N meilleurs individus pour former la population Parent POP_{t+1}
14: **Tant que**$| Q_t | < N/2$
15: Sélectionner deux parents P_1 et $P_2 \in POP_{t+1}$ par tournoi binaire
16: Créer 2 enfants Enf_1 et Enf_2 par croisement
17: Évaluer les enfants et mettre à jour A
18: Tirer aléatoirement un nombre *rnd* entre *0* et *1*
19: **Si** $rnd < p_m$ **alors**
20: Mute les enfants
21: **Fin si**
22: *Winner*= Compet (Enf_1 , Enf_2)
23: Ajouter *Winner* à Q_t
24: **Fin tant que**
25: Classer POP_{t+1} en front
26: Total = somme des facteurs d'isolement des individus non dominés
27 : **Pour** chaque individu x du 1er front **faire**
28 : nb_clones = $\lceil$ (Dist(x)* *N/2*)/Total $\rceil$
29 : cpt =0
30 : **Tant que** cpt< nb_clones
31 : (c_1, c_2)= Clone (x)
32 : c_1^{mut}= mutation_1(c_1)
33 : c_2^{mut}= mutation_2(c_2)
34 : Évaluer les clones mutés et mettre à jour A
35 : *Winner*= Compet (c_1^{hyp1}, c_2^{hyp2})
36: Ajouter *Winner* à Q_t
37: cpt = cpt +1
38: **Fin tant que**
39: **Fin pour**
40: $t = t + 1$
41: **Fin tant que**
42: **Retourner A**

Algorithme 7.2 : Procédure *Compet* (Individu x, Individu y)

2 : **Si** x domine y **alors**
3 : retourner x
4 : **Sinon**
5 : **Si** x est dominé par y **alors**
6 : retourner y
7 : **Sinon**
8 : **Si** Dist(x) > Dist(y) **alors**
9 : retourner x
10 : **Sinon**
11 : **Si** Dist(x) < Dist(y) **alors**
12 : retourner y
13 : **Sinon**
14 : choisir aléatoirement entre x et y et retourner l'individu choisi
15 : **Fin si**
16 **Fin si**
17 : **Fin si**
18 : **Fin si**

7.4 Le problème de sac à dos multi-objectifs en 0/1 (MOKP)

Le MOKP utilisé dans ce chapitre est celui proposé par Zitzler et Thiele [1999] et Zitzler *et al.* [2001] dans leurs expérimentations. Nous avons choisi ce problème car c'est l'un des problèmes les plus étudiés dans la communauté multi-objectifs. Ainsi, nous pourrons comparer les performances de notre approche avec celles des meilleurs algorithmes de la littérature.

Le *nobj*-objectifs problème de sac à dos en 0/1 est défini par un ensemble de J items (ou objets). À chacun des items sont associés Z valeurs de profit ainsi que Z poids. Formellement, le problème peut être exprimé de la manière suivante. Soit un ensemble de J objets et Z sacs à dos avec :

- $p_{z,j}$ - profit correspondant à l'objet j en fonction du sac à dos z,
- $w_{z,j}$ – poids correspondent à l'objet j en fonction du sac à dos z,
- cap_z - capacité totale du sac à dos z,

Il faut donc trouver un vecteur $x = \{x_1, x_2, \ldots, x_J\} \in \{0,1\}^J$ qui maximise

$$f_z(x) = \sum_{j=1}^{J} p_{z,j} x_j \qquad z = 1,\ldots, Z$$

sujet à :

$$\sum_{j=1}^{J} w_{z,j} x_j \leq cap_z \qquad z = 1,\ldots, Z$$

Le problème consiste donc à sélectionner un sous-ensemble d'objets x_j maximisant $f(x)=(f_1(x), f_2(x), \ldots, f_z(x))$ et pour lequel les contraintes de capacité sont respectées.

Comme pour sa version uni-objectif, le problème de sac à dos multi-objectifs peut être utilisé pour modéliser de nombreux problèmes réels, comme la répartition de budgets ou encore l'allocation de ressources. Plusieurs heuristiques ont été développées pour résoudre ce problème. En recensant la littérature, on retrouve, sans être exhaustif, des méthodes de recherche dans le voisinage comme le recuit simulé [Serafini 1994; Ulungu *et al.* 1999], la recherche avec Tabou [Gandibleux *et al.* 1996], des méthodes évolutionnaires comme le VEGA [Schaffer 1985], le NSGA [Srivinas et Deb 1994] et le NSGAII [Deb 2000], le SPEA [Zitzler et Thiele 1998] et le SPEA2 [Zitzler *et al.* 2001], le PMS^{MO} [Zinflou *et al.* 2006], ou encore le M-PAES [Knowles et Corne 2000]. Plus récemment, des approches hybrides combinant algorithme génétique et méthode de recherche locale ont été proposées pour résoudre le problème comme le *Multi-Objective Genetic Local Search* (MOGLS) [Jaszkiewicz 2000] ou le *Genetic Tabu Search for the Multi-Objective Knapsack Problem* (GTS^{MOKP}) [Barichard 2003]. Le MOGLS utilise une méthode de descente pure comme opérateur de recherche locale alors que le GTS^{MOKP}, de son côté, utilise un algorithme de recherche avec tabous comme opérateur de recherche locale. Ces deux algorithmes, contrairement à GISMOO, ne sont pas des approches Pareto et ils utilisent une fonction d'agrégation linéaire pondérée pour évaluer la performance des individus trouvés au cours des itérations. Mentionnons que parmi les approches hybrides pour le MOKP, le MOGLS est un des algorithmes les plus réputés et est souvent utilisé dans la littérature pour des fins de comparaison. Toutefois, à notre connaissance, les méthodes hybrides utilisées pour résoudre ce problème se limitent principalement à des approches hybridant algorithmes génétiques et méthodes de recherche locale.

7.4.1 Espace de recherche et représentation

Le MOKP utilisé dans cette partie de la thèse est un problème multi-objectifs en *0/1*. Une solution réalisable à ce problème est donc un vecteur binaire de n composantes satisfaisant les différentes contraintes du problème. Par conséquent, nous avons opté pour une représentation classique sous forme de chaîne de bits pour représenter chaque individu. L'espace de recherche S est constitué par l'ensemble de tous les vecteurs binaires, qui est clairement un sous-ensemble de $\{0, 1\}^n$.

7.4.2 Génération de la population initiale pour le MOKP

Comme mentionné à la Section 7.3, l'algorithme GISMOO commence en générant une population de départ constituée à 70% d'individus générés aléatoirement et à 30% d'individus générés en utilisant une heuristique gourmande. Pour construire chaque individu aléatoirement dans le cas du MOKP, on ajoute simplement à l'individu en cours de création un sous-ensemble d'objets pris au hasard. Si la somme des poids de tous les objets choisis aléatoirement excède la capacité d'un des sacs, on retire progressivement les objets influant le moins sur le profit de manière à respecter les contraintes de capacité. L'heuristique gourmande utilisée pour générer des individus est la même que celle proposée par Jaszkiewicz [2000] et consiste simplement à construire un individu x en insérant progressivement les objets influant le plus sur le profit tout en respectant les contraintes de capacité.

7.4.3 Opérateur de croisement pour le MOKP

Dans la version de GISMOO pour le MOKP, nous utilisons l'opérateur de croisement classique à un point de coupure [Holland 1962]. Notons que l'individu résultant d'un

croisement peut comporter beaucoup plus d'objets que chaque parent pris séparément. Par conséquent, il peut violer les contraintes de capacité du MOKP. Pour restaurer la faisabilité des individus créés par croisement, nous procédons de la même manière que lors de la création de la population initiale en retirant progressivement les objets influant le moins sur le profit.

7.4.4 Opérateur de mutation pour le MOKP

Pour résoudre le MOKP, nous utilisons deux types de mutation dans notre algorithme. Le premier opérateur de mutation est une inversion de bit simple. Cet opérateur de mutation consiste à inverser de manière aléatoire un gène de l'individu à muter (si le gène sélectionné est à 0, on le met à 1 et inversement). Le second opérateur de mutation est celui décrit par Knowles et Corne [2000]. Cet opérateur de mutation consiste à tirer un nombre aléatoire *nb* pour chaque gène du chromosome de l'individu à muter. Si le reste de la division entière de *nb* par le quart de la taille du chromosome est nul, on inverse la valeur du gène correspondant. Notons que seul le deuxième opérateur de mutation est utilisé dans la phase génétique de GISMOO alors que les deux sont appliqués dans la phase immune. Finalement, il est important de mentionner qu'après la mutation, la faisabilité des individus mutés est restaurée de la même manière que lors de la création de la population initiale.

7.5 Expérimentations numériques

L'algorithme GISMOO proposé dans ce chapitre a été implémenté en C++ avec Visual Studio .Net 2005. L'ordinateur utilisé pour les expérimentations numériques est un Dell

équipé d'un processeur Pentium Xeon 3.6 Ghz avec 1 Go de mémoire vive sous Windows XP.

7.5.1 Jeux d'essais

Les expérimentations numériques présentées dans ce chapitre sont réalisées à l'aide des neuf instances du problème de sac à dos multi-objectifs publiées par Zitzler et Thiele [1999]. Ces jeux d'essais ont 2, 3 et 4 objectifs combinés avec 250, 500 et 750 objets. De plus, pour chaque instance, le nombre de sacs est égal au nombre d'objectifs. Ces instances ont été construites aléatoirement par les auteurs. Il n'y a pas de corrélation entre les poids et les profits. De plus, les capacités totales de chaque sac sont réglées à environ la moitié de la somme des poids des objets pour le sac considéré. Il en découle que le nombre d'objets présents attendus dans les solutions optimales est d'environ la moitié du nombre total d'objets du problème [Zitzler et Thiele 1999].

7.5.2 Comparaison expérimentale

Dans cette section, nous comparons, dans un premier temps, GISMOO au PMS^{MO} et au NSGAII qui sont deux algorithmes Pareto de la littérature. Dans un deuxième temps, nous comparons les performances de notre approche à celles du MOGLS qui est un algorithme hybride réputé de la littérature comptant parmi les approches les plus performantes pour résoudre le MOKP. Ces trois algorithmes ont été choisis, d'une part, parce qu'ils illustrent bien l'état de l'art des algorithmes évolutionnaires en optimisation multi-objectifs et, d'autre part, leurs codes sources ont été rendus disponibles par leurs auteurs respectifs.
Comme pour GISMOO, les algorithmes PMS^{MO}, NSGAII et MOGLS ont été implémentés en C^{++}. Dans notre implémentation, les principales structures de données sont partagées par

tous les algorithmes. Nous obtenons ainsi une base de comparaison commune pour évaluer équitablement les performances des différents algorithmes. Dans toutes les expérimentations numériques effectuées dans cette partie, chaque instance du MOKP a été résolue à 30 reprises par chaque algorithme sur la même machine test. La taille N des populations est la même pour chaque algorithme et dépend du nombre d'objectifs et du nombre d'objets de l'instance à traiter. Le Tableau 7.1 indique la taille des populations pour tous les algorithmes en fonction du nombre d'objectifs et d'objets de l'instance.

Nombre d'objectifs	**Nombre d'objets**		
	250	**500**	**750**
2	150	200	250
3	200	250	300
4	250	300	400

Tableau 7.1 : Taille de la population selon le nombre d'objectifs et d'objets de l'instance à résoudre

7.5.2.1 Comparaison avec d'autres approches Pareto

Dans cette première série d'expérimentations numériques, nous allons comparer l'approche GISMOO avec les deux approches Pareto mentionnées à la section précédente, le NSGAII et le PMS^{MO}. Pour tous les problèmes testés dans cette partie, chaque algorithme a été exécuté pendant 500 générations. Pour les trois algorithmes, la probabilité de mutation a été fixée à 0.06. Notons ici que GISMOO ne nécessite pas de fixer une probabilité de croisement. Pour les deux autres algorithmes, la probabilité de croisement est fixée à 1.00. Finalement, la taille de l'archive locale du PMS^{MO} est fixée à 100. La valeur des différents paramètres du NSGAII et du PMS^{MO} a été fixée en accord avec les expérimentations effectuées par Zitzler *et al.*[2001] et par Zinflou *et al.*[2006].

Mentionnons aussi que, dans ce travail, nous ne comparons pas directement les performances du NSGAII par rapport à celles du PMS^{MO}. Le lecteur peut consulter [Zinflou *et al.* 2006] pour une comparaison directe entre les deux algorithmes.

Le Tableau 7.2 présente les résultats de la métrique d'hyper-volume ***H*** (présentée à la Section 3.8.3 du Chapitre 3) pour les trois algorithmes. Les deux premières colonnes du tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats moyens du NSGAII, du PMS^{MO} et de GISMOO pour la métrique ***H***. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. À partir des résultats du Tableau 7.2, nous observons que GISMOO obtient de meilleurs résultats que les deux autres algorithmes sur toutes les instances testées. En effet, la taille de l'espace dominé par GISMOO est plus importante que celle du NSGAII et du PMS^{MO}. On note toutefois que l'écart entre GISMOO et NSGAII est plus important qu'entre GISMOO et PMS^{MO}.

Nombre d'objectifs	Nombre d'objets	NSGAII	PMS^{MO}	GISMOO
2	250	9.47 E+7	9.84 E+7	9.86 E+7
	500	3.95 E+8	4.02 E+8	4.07 E+8
	750	8.57 E+8	8.66 E+8	8.93 E+8
3	250	8.06 E+11	9.03 E+11	9.28 E+11
	500	6.42 E+12	7.04 E+12	7.70 E+12
	750	2.26 E+13	2.45 E+13	2.71 E+13
4	250	6.36 E+15	7.13 E+15	7.94 E+15
	500	1.01 E+17	1.08 E+17	1.34 E+17
	750	5.19 E+17	5.47 E+17	7.15 E17

Tableau 7.2 : Moyenne de l'hyper-volume ***H*** du NSGAII, PMS^{MO} et GISMOO pour les instances du problème de sac à dos multi-objectifs

Si l'hyper-volume donne une bonne idée de la taille de l'espace dominé par un ensemble de solutions, cette métrique ne permet pas de comparer deux ensembles de solutions l'un

par rapport à l'autre. Pour y parvenir, nous utilisons la métrique ***C*** (décrite à la Section 3.8.2 du Chapitre 3). La Figure 7.1 présente les résultats des trois algorithmes selon la métrique ***C*** suivant la présentation adoptée dans [Zitzler et Thiele 1999]. Dans cette figure, chaque boîte correspond à la comparaison des solutions de l'algorithme *A* en ligne avec l'algorithme *B* en colonne. Ainsi, la valeur ***C***(A,B) indique le pourcentage d'éléments de *B* dominés par au moins un élément de *A*. Pour chaque boîte, l'échelle part de 0 en bas de la boîte pour aller à 1 en haut de celle-ci. Chaque boîte contient 9 graphiques correspondants respectivement, de gauche à droite, aux instances contenant 250, 500 et 750 objets regroupées en fonction du nombre d'objectifs (2, 3 et 4). Chaque barre horizontale noire correspond à la moyenne des différentes mesures ***C*** pour 30 exécutions et les barres verticales indiquent l'écart entre les valeurs maximales et minimales trouvées lors des différentes exécutions.

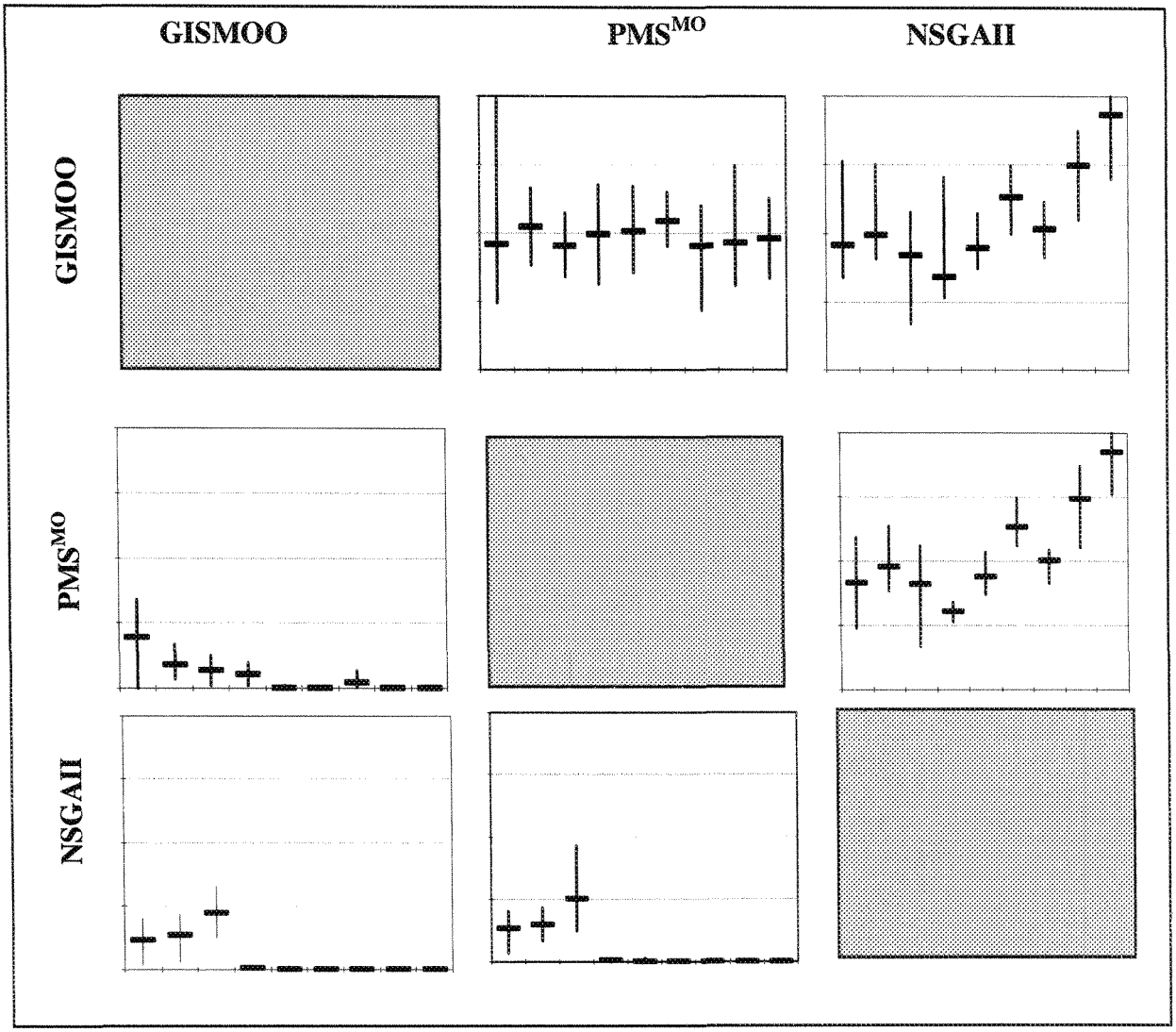


Figure 7.1 : Couverture moyenne C de GISMOO, PMS^{MO} et NSGAII pour le problème de sac à dos multi-objectifs

Les résultats obtenus avec la métrique C confirment les résultats obtenus avec l'hyper-volume, à savoir une meilleure performance de GISMOO comparativement au NSGAII et au PMS^{MO}. En effet, les valeurs de $C(GISMOO, PMS^{MO})$ et $C(GISMOO, NSGAII)$ sont respectivement supérieures à celles de $C(PMS^{MO}, GISMOO)$ et $C(NSGAII, GISMOO)$. On note aussi que la différence entre l'approche proposée dans ce chapitre et les deux

autres algorithmes Pareto semble augmentée en fonction du nombre d'objectifs. En effet, à l'exception des problèmes à deux objectifs, le rapport entre les résultats C(GISMOO, PMS^{MO}) et C(GISMOO, NSGAII) par rapport à C(PMS^{MO}, GISMOO) et C(NSGAII, GISMOO) est supérieur à 10 ce qui signifie que la grande majorité des solutions trouvées par GISMOO sont de qualité égale ou supérieure à celles du PMS^{MO} et du NSGAII. Toutefois, les résultats de la métrique ***C*** ne permettent pas de déterminer si les solutions d'un algorithme sont mieux distribuées que celle d'un autre.

Pour confirmer les résultats obtenus avec la métrique ***C*** et avoir une idée de la distribution des solutions dans l'espace de recherche, nous avons aussi comparé les trois algorithmes en utilisant la métrique ***Diff***, présentée à la Section 3.8.4 du Chapitre 3, qui permet de mesurer la différence de couverture entre les solutions de deux algorithmes. Les Tableaux 7.3 et 7.4 résument respectivement les résultats obtenus entre GISMOO et PMS^{MO} et entre GISMOO et NSGAII en utilisant cette métrique. Les deux premières colonnes de chaque tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats moyens obtenus pour la métrique ***Diff*** après 30 exécutions indépendantes de chacun des algorithmes à comparer. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris.

Nombre d'objectifs	Nombre d'objets	PMS^{MO}	GISMOO
2	250	1.96 E+4	2.07 E+5
	500	1.70 E+6	5.02 E+6
	750	3.99 E+4	2.74 E+7
3	250	4.83 E+8	2.56 E+10
	500	1.78 E+7	6.55 E+11
	750	1.23 E+7	2.63 E+12
4	250	1.67 E+12	8.13 E+14
	500	1.73 E+11	2.60 E+16
	750	3.31 E+8	1.68 E+17

Tableau 7.3 : Différence de couverture moyenne du PMS^{MO} et GISMOO pour les instances du problème de sac à dos multi-objectifs

Nombre d'objectifs	Nombre d'objets	NSGAII	GISMOO
2	250	1.85 E+4	3.94 E+6
	500	4.84 E+4	1.29 E+7
	750	5.48 E+4	3.56 E+7
3	250	1.77 E+8	1.22 E+11
	500	1 E+9	1.14 E+12
	750	1.77 E+9	4.27 E+9
4	250	2.49 E+12	1.58 E+15
	500	1.76 E+13	3.33 E+16
	750	3.14 E +13	1.96 E+17

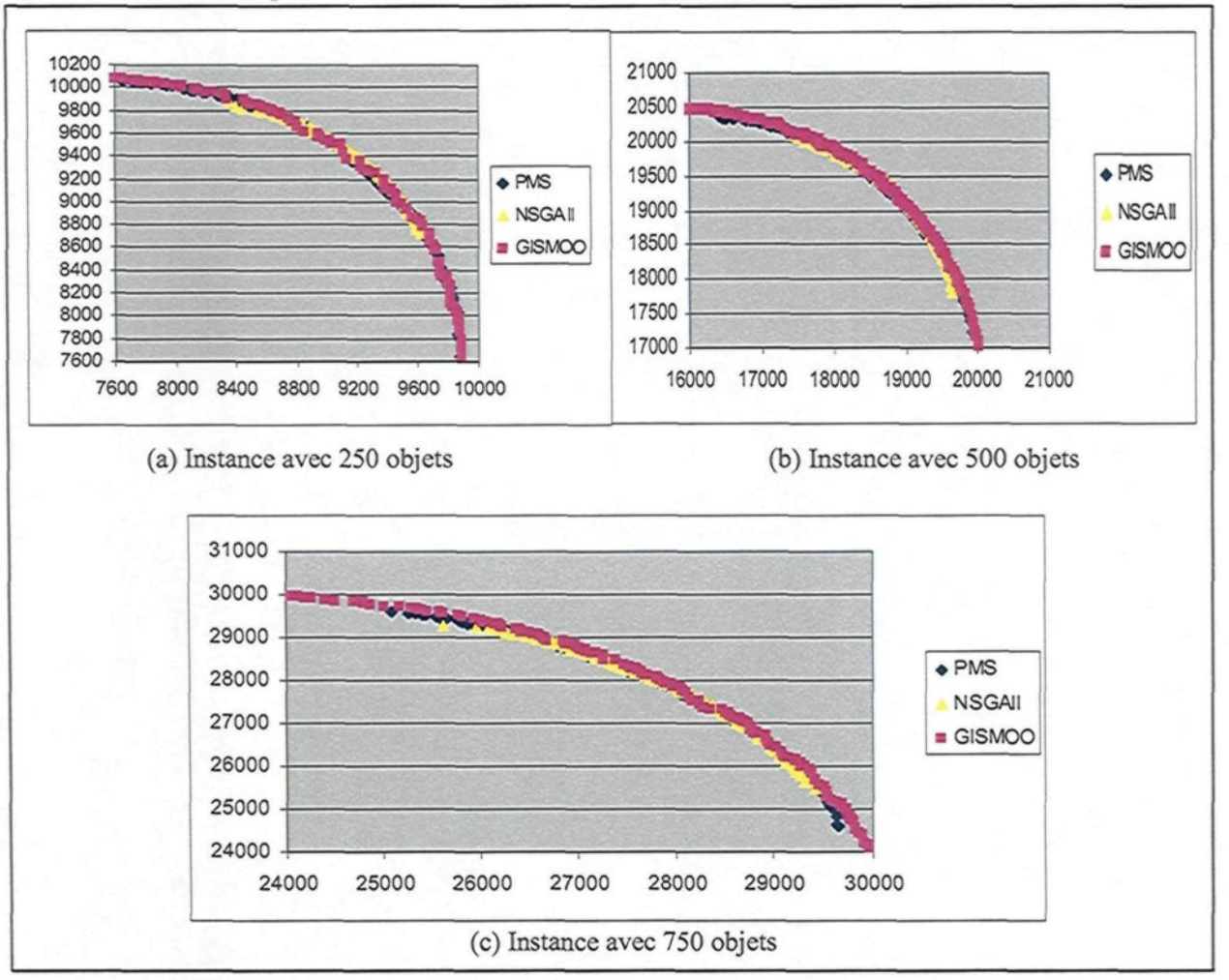
Tableau 7.4 : Différence de couverture moyenne du NSGAII et GISMOO pour les instances du problème de sac à dos multi-objectifs

En examinant les deux tableaux, on note, une fois de plus, que les résultats sont nettement en faveur de GISMOO. En effet, lorsqu'on compare, dans un premier temps, GISMOO au PMS^{MO} (Tableau 7.3), on note que GISMOO obtient une plus grande différence de couverture que le PMS^{MO} pour toutes les instances. On note aussi que l'écart entre les deux algorithmes semble être en relation avec la taille de l'instance et le nombre d'objectifs. Ainsi, l'écart de couverture entre les deux algorithmes augmente jusqu'à 10^9 pour le problème de 750 objets et 4 objectifs. On remarque aussi que malgré la différence de couverture entre les deux algorithmes et les résultats observés avec l'hyper-volume, les

deux algorithmes semblent explorer différentes régions de l'espace des objectifs. Cette situation s'explique sans doute par le fait que les deux algorithmes utilisent différents mécanismes d'exploration. En comparant, dans un deuxième temps, GISMOO et le NSGAII (Tableau 7.4), on remarque aussi un net avantage en faveur de GISMOO. En effet, GISMOO obtient une plus grande différence de couverture que le NSGAII sur toutes les instances testées. Toutefois, contrairement à ce que l'on avait observé au Tableau 7.3, l'écart entre les deux algorithmes ne semble pas être en relation avec le nombre d'objectifs, mais plutôt avec la taille des instances. Ces résultats laissent supposer que les solutions trouvées par GISMOO ont une meilleure distribution que celles trouvées par le NSGAII et le PMS^{MO}.

Pour nous en convaincre, nous avons représenté graphiquement les solutions des trois algorithmes pour les instances à deux objectifs. La Figure 7.2 (a, b, c) montre les ensembles Pareto obtenues par chacun des trois algorithmes pour les trois instances bi-objectives. Cette représentation graphique confirme les résultats des différentes mesures. Pour les trois instances, il apparaît clairement que l'ensemble Pareto proposé par le NSGAII est plus étriqué que celui proposé par GISMOO. En effet, les points des ensembles Pareto du NSGAII sont, pour la plupart, tous regroupés dans une même région de l'espace de recherche. Cette situation explique les résultats observés avec la métrique ***Diff***. En effet, les mécanismes d'exploration de GISMOO lui permettent d'obtenir une meilleure dispersion des solutions dans l'espace de recherche alors que le NSGAII semble obtenir une meilleure exploration d'une région particulière de cet espace. Pour l'instance avec 250 objets il est très difficile de différencier les ensembles proposés par GISMOO et le PMS^{MO} car les

courbes sont pratiquement confondues. Par contre, pour les deux autres instances, on remarque clairement que l'ensemble Pareto de GISMOO est plus étendu que celui calculé par le PMS^{MO}. On constate donc, une fois de plus, que GISMOO explore mieux l'espace de solutions que le PMS^{MO}.



(a) Instance avec 250 objets

(b) Instance avec 500 objets

(c) Instance avec 750 objets

Figure 7.2 : Exemple de surfaces de compromis trouvées par le PMS^{MO}, le NSGAII et GISMOO pour les trois instances à deux objectifs

Le Tableau 7.5 indique (en secondes), pour chacun des trois algorithmes et pour chaque instance du problème, les temps moyens d'exécution obtenus. On note, à l'aide de ce tableau, que GISMOO nécessite toujours moins de temps de calcul que le NSGAII et le PMS^{MO}.

Nombre d'objectifs	Nombre d'objets	NSGAII	PMS^{MO}	GISMOO
2	250	15	21	9
	500	20	41	13
	750	43	94	24
3	250	25	73	19
	500	53	89	35
	750	75	121	55
4	250	64	100	52
	500	103	150	85
	750	205	302	108

Tableau 7.5 : Moyenne des temps de calculs (secondes) du NSGAII, PMS^{MO} et GISMOO pour les instances du problème de sac à dos multi-objectifs

En complément, nous avons voulu comparer l'apport de chacune des deux phases sur la performance globale de GISSMMOO. La Figure 7.3 (a, b et c) compare graphiquement les ensembles de solutions obtenus par GISMOO, par GISMOO sans phase immune (Phase génétique) et par GISMOO sans phase génétique (Phase immune) pour les trois instances bi-objectives. À partir de cette figure, on note que les courbes proposées par chacune des deux phases de GISMOO exécutées séparément sont nettement dominées par l'ensemble Pareto trouvé par GISMOO. On note aussi, en comparant les deux phases l'une par rapport à l'autre, que la phase génétique effectue une meilleure intensification de la recherche tandis que la phase immune semble être capable d'effectuer une meilleure diversification. En effet, les courbes proposées par la phase immune sont plus étendues que celles de la phase génétique. Toutefois, ces courbes ne sont pas continues. À l'opposé, les courbes

proposées par la phase génétique sont continues, mais généralement plus étriquées. Les résultats de cette figure mettent en évidence la complémentarité des deux phases et expliquent les bons résultats de GISMOO.

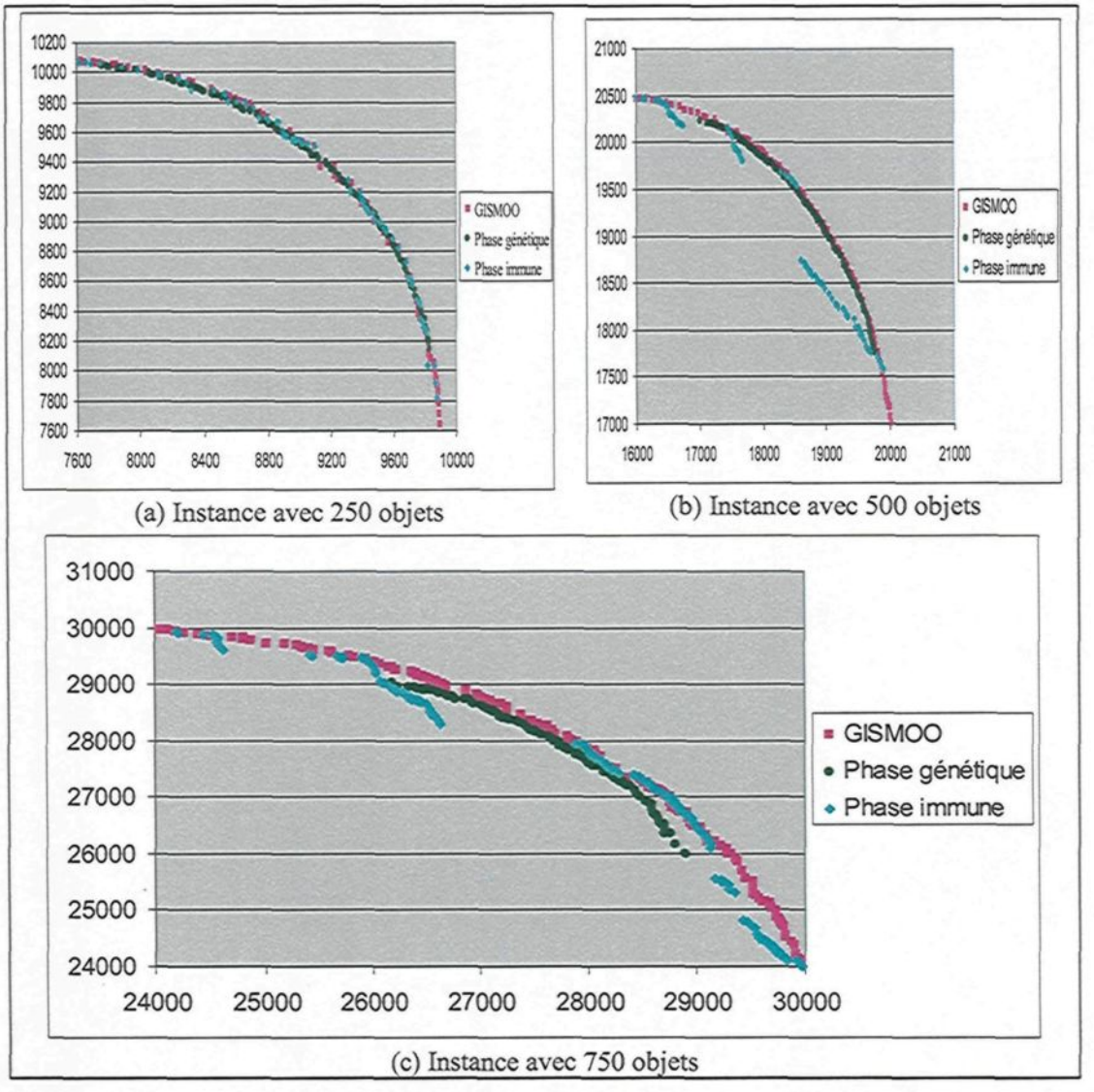


Figure 7.3 : Apport des différentes phases sur la performance globale de GISMOO

Les résultats précédents mettent en évidence l'efficacité de l'approche hybride proposée dans ce chapitre pour le problème de sac à dos multi-objectifs en 0/1. En particulier, GISMOO semble être capable d'identifier les régions moins explorées dans l'espace de solutions afin d'orienter la recherche vers ces régions. Ainsi, comparativement aux deux autres algorithmes Pareto utilisés dans cette partie, GISMOO parvient à proposer des ensembles Pareto avec des solutions mieux distribuées dans l'espace de recherche. Dans la pratique, les ensembles de solutions proposés par GISMOO sont plus intéressants pour un décideur que ceux du NSGAII ou du PMS^{MO} car ils contiennent des solutions diversifiées. Ainsi, les ensembles Pareto trouvés par GISMOO facilitent la prise de décision du gestionnaire en lui offrant une vision plus large et plus diversifiée des solutions potentielles à adopter.

Toutefois, les résultats obtenus laissent aussi entrevoir des possibilités de gains en terme d'intensification de la recherche dans certaines régions. Pour nous en convaincre, nous allons comparer les performances de GISMOO avec celle d'un algorithme non Pareto hybridant un algorithme génétique et une procédure de recherche locale : le MOGLS. Cet algorithme est réputé dans la littérature pour ses capacités d'intensification [Jaszkiewicz 2000] et compte aussi parmi les approches les plus performantes pour résoudre le MOKP [Barichard 2003].

7.5.2.2 Comparaison avec un algorithme non Pareto

Dans les expérimentations numériques suivantes, nous avons comparé GISMOO avec le MOGLS de Jaszkiewicz [2000] et une version de GISMOO avec recherche locale (GISMOO+LS). L'ajout d'une procédure de recherche locale à notre algorithme a pour but,

d'une part, de mesurer l'apport éventuel d'un mécanisme explicite d'intensification aux performances de notre approche. D'autre part, plusieurs auteurs [Knowles et Corne 2000; Berro 2001] ont relevé la difficulté à comparer des algorithmes mimétiques multi-objectifs avec des approches Pareto classiques du fait des différences de fonctionnement entre les deux type d'approches (nombre d'évaluations, nombre de générations, etc...). L'ajout de la même procédure de recherche locale utilisée par le MOGLS à GISMOO nous permettra de mieux comparer les mécanismes d'explorations des deux algorithmes. La méthode de recherche locale implémentée dans cette partie est une méthode de descente simple [Jaszkiewicz 2000]. Ainsi, à chaque itération de l'algorithme, on choisit aléatoirement un individu de la population Q et cet individu est amélioré localement dans 50% des cas. On note que, dans l'algorithme du MOGLS, la solution générée par croisement est améliorée à chaque itération. Il est important de préciser que, contrairement aux algorithmes présentés dans la section précédente, le critère d'arrêt du MOGLS ne correspond pas à un nombre maximum de générations. En effet, pour les algorithmes agrégatifs comme le MOGLS, la notion de génération n'est présente que pour garder la présentation classique des algorithmes évolutionnaires [Barichard 2003]. Dans ces expérimentations, Jaszkiewicz [2000] note que 50 générations du MOGLS représentent approximativement 500 générations d'un AG multi-objectifs standard comme le NSGA ou le SPEA. Plus récemment, Barichard [2003] utilisa le même nombre de générations pour comparer son algorithme agrégatif au NSGA. Conformément à ces deux expérimentations, nous avons fixé le nombre de générations du MOGLS à 50 de manière à être le plus équitable possible. Pour ce qui est de la taille de la population, elle est choisie en fonction du problème à

résoudre selon les valeurs du Tableau 7.1. Notons aussi que nous avons limité le nombre maximum de générations à 250 pour la version de GISMOO avec recherche locale. En diminuant ainsi le nombre de générations de GISMOO+LS, nous tenons compte des évaluations supplémentaires causées par l'addition de la procédure de recherche locale de manière à avoir approximativement le même nombre d'évaluations pour les deux versions de notre algorithme.

Le Tableau 7.6 présente les résultats de l'hyper-volume pour les trois algorithmes. Les deux premières colonnes du tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats moyens de 30 exécutions du MOGLS, de GISMOO et de GISMOO+LS pour la métrique H. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris. À l'aide de ce tableau, on remarque, dans un premier temps, en comparant GISMOO et MOGLS, que les performances des deux algorithmes sont très proches l'une de l'autre sur la plupart des problèmes avec un avantage pour le MOGLS. On remarque aussi que, pour les instances à trois objectifs, l'écart entre les deux approches semble diminuer lorsque la taille de l'instance augmente. Lorsqu'on analyse maintenant l'effet de l'addition d'une procédure de recherche locale à l'algorithme GISMOO, on note que cet ajout améliore les performances de l'approche sur toutes les instances. En effet, à l'exception des instances à deux objectifs et contenant respectivement 250 et 750 objets où les trois algorithmes obtiennent les mêmes résultats moyens, GISMOO+LS obtient les meilleurs résultats sur toutes les autres instances. Ainsi, on note que l'ajout d'une procédure de recherche locale à notre approche permet d'obtenir des résultats supérieurs à ceux du MOGLS pour 5

instances tout en obtenant des résultats moyens identiques pour les quatre instances restantes.

Nombre d'objectifs	Nombre d'objets	MOGLS	GISMOO	GISMOO + LS
2	250	9.86 E+7	9.86 E+7	9.86 E+7
	500	4.08 E+8	4.07 E+8	4.08 E+8
	750	8.93 E+8	8.93 E+8	8.93 E+8
3	250	9.33 E+11	9.28 E+11	9.34 E+11
	500	7.74 E+12	7.70 E+12	7.74 E+12
	750	2.72 E+13	2.71 E+13	2.73 E+13
4	250	8.09 E+15	7.94 E+15	8.12 E+15
	500	1.36 E+17	1.34 E+17	1.37 E+17
	750	7.19 E+17	7.15 E17	7.29 E+17

Tableau 7.6 : moyenne de l'hyper-volume ***H*** du MOGLS, GISMOO et GISMOO +LS pour les instances du problème de sac à dos multi-objectifs

Les faibles écarts constatés entre GISMOO et MOGLS combinés aux résultats obtenus avec l'ajout d'une méthode d'intensification à notre approche hybride semblent confirmer les lacunes en terme d'intensification de la recherche de notre algorithme observées à la section précédente. Toutefois, ces résultats laissent supposer que, même si GISMOO n'intensifie pas la recherche comme le MOGLS, il parvient à trouver des solutions dans toutes les régions de l'espace de solutions ce qui explique sans doute les faibles écarts en terme d'hyper-volume entre les deux algorithmes.

La Figure 7.4 présente les résultats des trois algorithmes selon la métrique ***C*** de façon similaire à la Figure 7.1. Ainsi, chaque boîte de la figure correspond à la comparaison des solutions de l'algorithme *A* en ligne avec l'algorithme *B* en colonne. On observe, à partir de cette figure, que les résultats de GISMOO sont inférieurs à ceux du MOGLS et de GISMOO+LS pour toutes les instances. En comparant maintenant GISMOO+LS avec

MOGLS, on note qu'à l'exception des trois instances à deux objectifs, les résultats penchent en faveur de GISMOO+LS. Ainsi, GISMOO+LS obtient de meilleurs résultats que le MOGLS pour 5 instances, est inférieur pour trois instances tout en obtenant un résultat identique pour l'instance restante. En examinant les graphiques de la Figure 7.4, on remarque aussi que l'avantage de GISMOO+LS par rapport au MOGLS augmente en fonction du nombre d'objectifs et du nombre d'objets du problème à résoudre. Ces résultats laissent supposer que les mécanismes d'exploration de GISMOO+LS sont plus performants que ceux du MOGLS, en particulier lorsque le nombre d'objets et le nombre d'objectifs du problème à résoudre augmentent. Toutefois, pour les problèmes bi-objectifs, le MOGLS effectue une meilleure intensification de la recherche comparativement à GISMOO+LS.

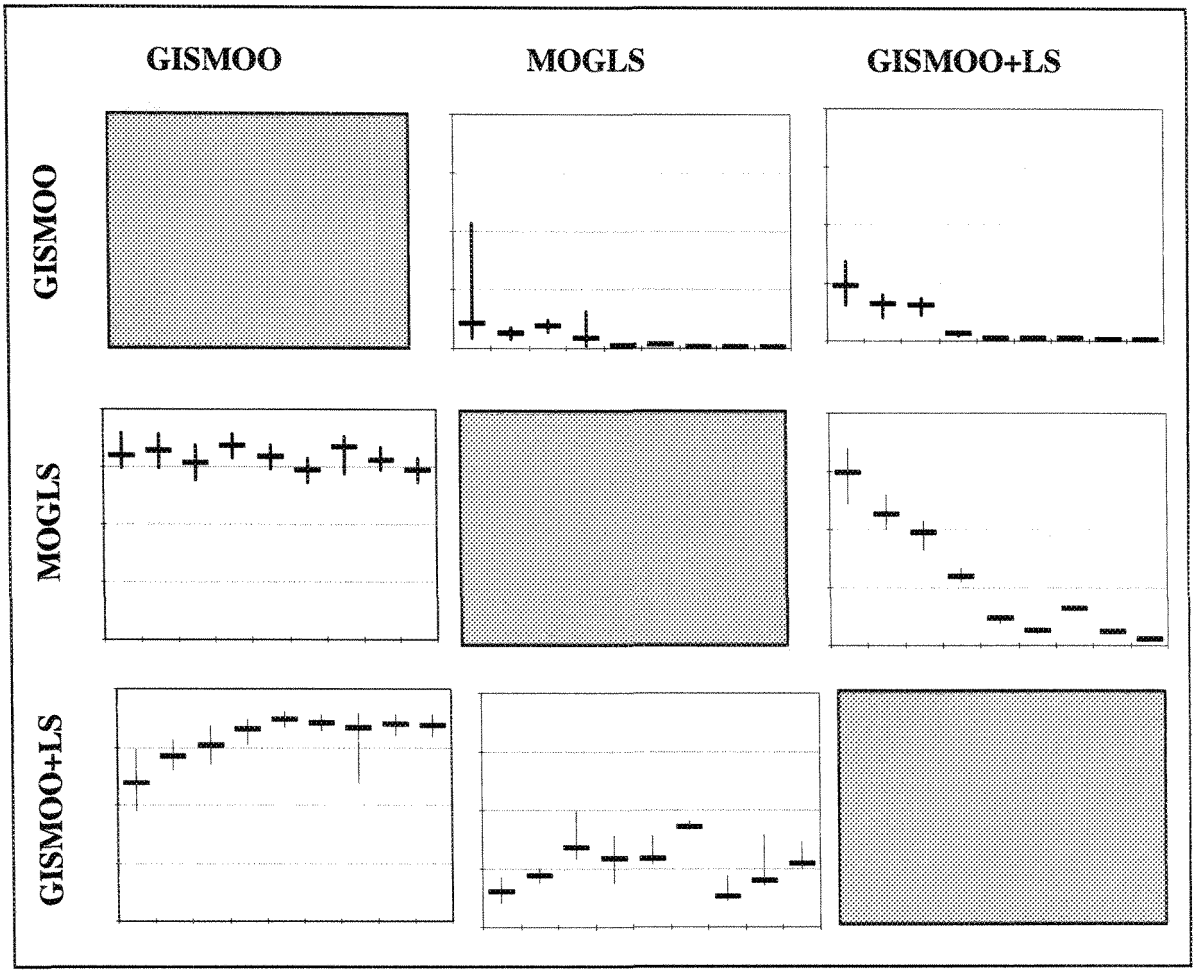


Figure 7.4 : Couverture moyenne C de GISMOO, MOGLS et GISMOO+LS pour le problème de sac à dos multi-objectifs

Pour confirmer les tendances observées à l'aide de la métrique C, nous avons également comparé les trois algorithmes en utilisant la métrique ***Diff*** qui permet de mesurer la différence de couverture entre les solutions de deux algorithmes. Les Tableaux 7.7 à 7.9 résument respectivement les résultats obtenus entre GISMOO et MOGLS, entre GISMOO et GISMOO+LS et entre GISMOO+LS et MOGLS en utilisant cette métrique. Les deux premières colonnes de chaque tableau indiquent respectivement le nombre d'objectifs et le nombre d'objets du problème à résoudre. Les colonnes suivantes montrent les résultats

moyens de 30 exécutions indépendantes de chacun des algorithmes pour la métrique ***Diff***. En ce qui concerne les meilleurs résultats obtenus, ils sont indiqués avec une trame en gris.

Nombre d'objectifs	Nombre d'objets	GISMOO	MOGLS
2	250	3.23 E+3	7.60 E+4
	500	4.25 E+4	2.16 E+5
	750	9.94 E+3	3.26 E+5
3	250	7.58 E+7	5.67 E+9
	500	7.37 E+8	3.94 E+10
	750	4.50 E+9	1.10 E+11
4	250	3.09 E+12	1.50 E+14
	500	4.87 E+13	2.23 E+15
	750	3.98 E+14	9.52 E+15

Tableau 7.7 : différence de couverture moyenne entre GISMOO et MOGLS pour les instances du problème de sac à dos multi-objectifs

Nombre d'objectifs	Nombre d'objets	GISMOO	GISMOO+LS
2	250	1.15 E+04	4.46 E+04
	500	4.70 E+4	1.61 E+5
	750	7.99 E+3	3.05 E+5
3	250	2.45 E+7	5.88 E+9
	500	1.01 E+8	4.66 E+10
	750	3.51 E+8	1.47 E+11
4	250	7.54 E+11	1.85 E+14
	500	4.62 E+12	2.98 E+15
	750	2.52 E+13	1.39 E+16

Tableau 7.8 : différence de couverture moyenne entre GISMOO et GISMOO+LS pour les instances du problème de sac à dos multi-objectifs

Nombre d'objectifs	Nombre d'objets	MOGLS	GISMOO+LS
2	250	4,36 E+4	4,03 E+3
	500	1.06 E+5	4.53 E+4
	750	5.42 E+4	3.58 E+4
3	250	5.78 E+08	8.56 E+8
	500	1.01 E+9	9.03 E+9
	750	1.25 E +9	4.3 E+10
4	250	6.54 E+12	3.97 E+13
	500	2.73 E+13	8 E+14
	750	8.99 E+13	4.39 E+15

Tableau 7.9 : différence de couverture moyenne entre le MOGLS et GISMOO +LS pour les instances du problème de sac à dos multi-objectifs

Les résultats obtenus avec la métrique ***Diff*** confirment les tendances observées précédemment. Ainsi, GISMOO+LS et le MOGLS obtiennent une meilleure différence de couverture que GISMOO pour toutes les instances (Tableaux 7.7 et 7.8). On note aussi que les écarts entre GISMOO et GISMOO+LS sont plus importants que ceux entre GISMOO et MOGLS pour les problèmes avec 3 et 4 objectifs. Par contre, pour les problèmes à 2 objectifs, l'écart entre GISMOO et MOGLS est plus important que celui entre GISMOO et GISMOO+LS. Lorsque l'on compare GISMOO+LS au MOGLS à l'aide du Tableau 7.9, on note que GISMOO+LS obtient les meilleurs résultats pour six des neuf instances et est inférieur seulement pour les instances à deux objectifs. Comme pour la métrique **C**, on observe que l'avantage de GISMOO+LS par rapport au MOGLS augmente en fonction du nombre d'objectifs et du nombre d'objets du problème à résoudre. Ainsi, pour les instances avec deux objectifs, l'écart entre les deux algorithmes a tendance à se réduire en fonction de l'augmentation du nombre d'objets. Pour les instances à trois et à quatre objectifs, l'écart entre les deux algorithmes à tendance à augmenter en relation avec l'augmentation du nombre d'objets. Ces résultats confirment que GISMOO+LS effectue une meilleure

exploration de l'espace de recherche que le MOGLS, en particulier lorsque le nombre d'objectifs et le nombre d'objets du problème augmentent. De plus, les mesures obtenues à l'aide de la métrique ***Diff*** nous permettent aussi de conclure que les surfaces de compromis calculées par GISMOO+LS sont généralement mieux distribuées que celles du MOGLS, en particulier pour les instances avec plus de deux objectifs.

La Figure 7.5 (a, b, c) représente graphiquement les surfaces des compromis générées par les trois algorithmes pour chacune des instances à deux objectifs. Cette représentation confirme les résultats obtenus avec les autres mesures. On note en observant les graphiques qu'il est très difficile de départager visuellement les trois algorithmes, car les surfaces de compromis sont quasiment confondues. On note toutefois que, pour l'instance à 250 objets, la surface de compromis du MOGLS est plus dense que celle de GISMOO et GISMOO+LS. Par contre, pour les deux autres instances, on observe que les surfaces de GISMOO et GISMOO+LS semblent légèrement plus étendues que celle du MOGLS. On note aussi que les solutions de GISMOO sont légèrement inférieures de celles fournies par MOGLS et GISMOO+LS. Ces observations expliquent les faibles écarts observés précédemment entre les performances des différents algorithmes sur les différentes mesures.

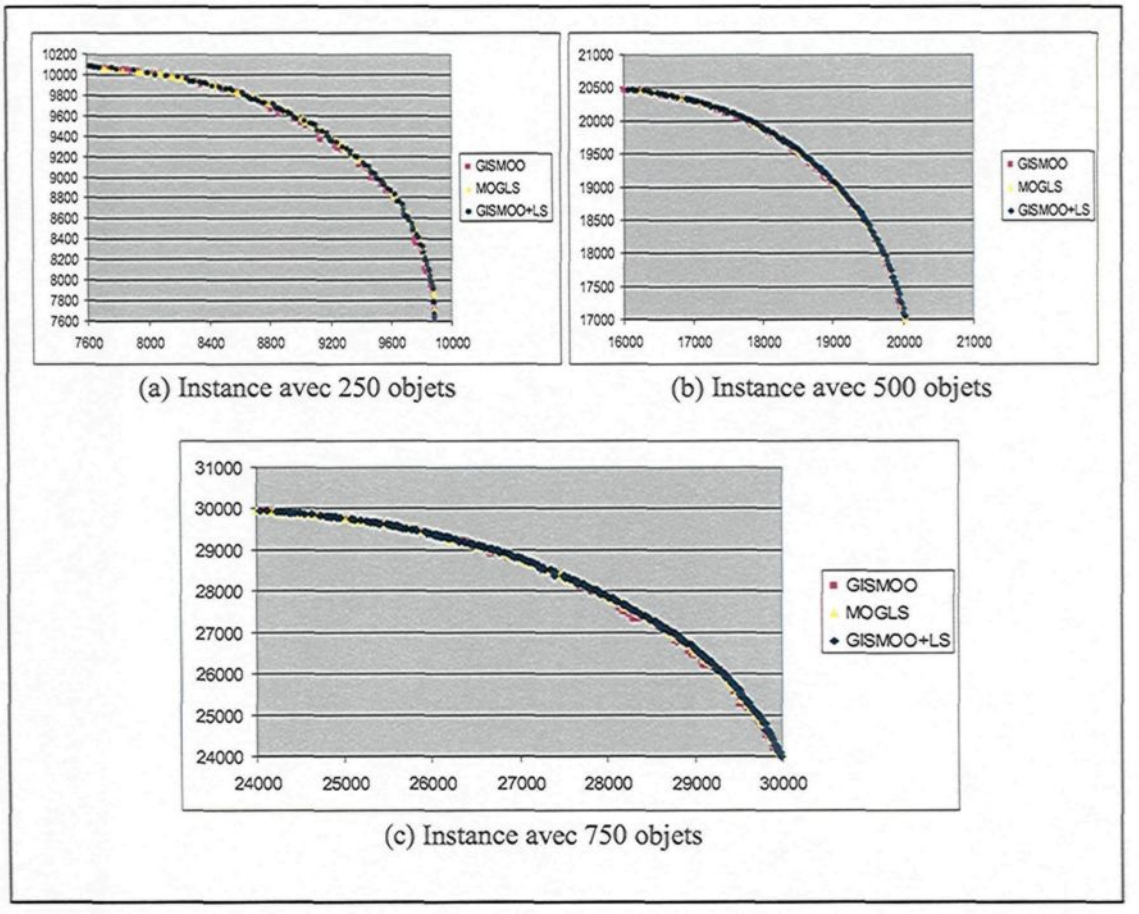


Figure 7.5 : Exemple de surfaces de compromis trouvées par GISMOO, MOGLS et GISMOO+LS pour les trois instances à deux objectifs

Le Tableau 7.10 indique (en secondes), pour chacun des trois algorithmes et pour chaque instance du problème, les temps moyens d'exécution obtenus. On note, à l'aide de ce tableau, que GISMOO nécessite toujours moins de temps de calcul que MOGLS qui lui nécessite moins de temps de calcul que GISMOO+LS. On note, ainsi que même si le nombre de générations de MOGLS est fixé à 50, le temps de calcul nécessaire à cet algorithme pour les réaliser est toujours supérieur à celui de GISMOO. Finalement, on remarque que l'ajout de la procédure de recherche locale à GISMOO entraîne une

augmentation du temps d'exécution de l'algorithme supérieure à 100% pour toutes les instances. Ces résultats sont intéressants sachant qu'à chaque itération le MOGLS n'effectue qu'un seul croisement. En effet, cela implique que le nombre d'opérations nécessaires pour effectuer ce croisement ainsi que le nombre d'amélioration locale effectuées par l'algorithme sont très important. Ceci explique pourquoi l'auteur de cet algorithme [Jaszkiewicz 2000] compare que 50 générations du MOGLS à 500 générations d'un algorithme évolutionnaire classique. Il est important de mentionner que les caractéristiques du MOKP font qu'il n'est pas nécessaire de réévaluer au complet une solution après chaque mouvement de recherche locale. L'écart entre les temps de calculs de GISMOO et du MOGLS serait donc encore plus important pour un problème ou une évaluation complète de la solution serait nécessaire après chaque mouvement.

Nombre d'objectifs	Nombre d'objets	MOGLS	GISMOO	GISMOO+LS
2	250	10	9	34
	500	19	13	48
	750	27	24	67
3	250	22	19	51
	500	53	35	97
	750	77	55	131
4	250	56	52	99
	500	107	85	177
	750	202	108	289

Tableau 7.10 : Moyenne des temps de calculs (secondes) du MOGLS, GISMOO et GISSMO+LS pour les instances du problème de sac à dos multi-objectifs

7.6 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme multi-objectifs Pareto (GISMOO) combinant des concepts issus des algorithmes génétiques avec des propriétés inspirées des systèmes immunitaires artificiels. Les expérimentations numériques

effectuées ont permis de démontrer l'efficacité de notre approche sur le problème de sac à dos multi-objectifs en 0/1. En effet, les performances de notre algorithme sont supérieures à celles obtenues par deux autres algorithmes Pareto représentatifs de l'état de l'art en optimisation multi-objectifs à l'aide d'algorithmes évolutionnaires : le NSGAII et le PMS^{MO}. De plus, les résultats expérimentaux montrent aussi que, lorsque l'on utilise des méthodes d'intensification équivalentes, les performances de GISMOO se comparent avantageusement par rapport à celles d'un algorithme mimétique comme le MOGLS. Une conclusion naturelle à ces expérimentations numériques est que, malgré sa simplicité, GISMOO est une alternative efficace pour résoudre des problèmes d'optimisation multi-objectifs. En effet, contrairement au NSGAII, au PMS^{MO} ou au MOGLS qui sont basés sur des modèles évolutionnaires complexes, GISMOO est un algorithme très simple à implémenter. En particulier, ces résultats mettent en évidence l'importance des mécanismes de diversification de la recherche sur l'efficacité globale d'un algorithme multi-objectifs. Les résultats de GISMOO sur ce benchmark classique de la littérature ont permis d'apprécier ses performances. Il est important de rappeler que GISMOO est un algorithme Pareto générique, ce qui signifie qu'il peut être adapté à la résolution de n'importe quel problème multi-objectifs.