PRTS+D et MEMOTS : nouvelles métaheuristiques pour l'optimisation combinatoire multicritère

T. Lust and J. Teghem

Faculté Polytechnique de Mons
Service de Mathématique et de Recherche Opérationnelle
9, rue de Houdain 7000 Mons (Belgique)
thibaut.lust@fpms.ac.be

Résumé Nous présentons dans cet article deux nouvelles métaheuristiques pour la résolution de problèmes d'optimisation combinatoire multicritère : la méthode PRTS+D, basée sur la recherche tabou et la méthode MEMOTS, fondée sur un algorithme mémétique. Ces deux méthodes n'utilisent pas de fonction d'agrégation basée sur des jeux de poids, mais une fonction d'évaluation basée sur le rang de Pareto. Nous comparons les résultats de PRTS+D et MEMOTS avec plusieurs métaheuristiques multicritères sur le problème du sac à dos multidimensionnel multiobjectif. Nous montrons que la méthode MEMOTS est très compétitive, puisque la méthode donne globalement de meilleures performances.

Mots-Clefs. Multiobjectif ; Recherche Tabou ; Algorithme Mémétique.

1 Introduction

Les problèmes d'optimisation sont fréquemment rencontrés par le chercheur et l'ingénieur et occupent une place de plus en plus importante à tous les niveaux de la société (conception de réseaux, tournée de véhicules, ordonnancement de tâches, construction d'horaires...). Dans chacun de ces problèmes, on définit un ou plusieurs critères (une consommation, un rendement, un coût, la faisabilité technologique, une durée de développement, l'impact sur l'environnement, etc.) correspondant à différentes fonctions à minimiser ou maximiser. On recherche ensuite des solutions optimales selon ces différents critères, vérifiant bien entendu les contraintes du problème.

Dans le cas d'un seul critère, il est facile de déterminer si une solution est meilleure qu'une autre, il suffit que la solution obtienne une meilleure évaluation au regard du critère défini. Or, il est fréquent que plusieurs critères interviennent dans la décision, et dans ce cas, modéliser un problème en utilisant un seul critère donne une solution biaisée et non satisfaisante. Dans le cas multicritère, une difficulté supplémentaire apparaît cependant, puisque les critères considérés sont généralement contradictoires ce qui conduit à des situations où améliorer un critère ne peut se faire qu'au détriment d'autres.

On définit un problème (P) d'optimisation combinatoire multicritère à K objectifs, n variables binaires et m contraintes de la façon suivante :

$$\left[\begin{array}{ll} \ll \min_{X \in D} \gg & z_k(X) = c^k X \qquad k = 1, \ldots, K \\ \text{où} & D = \{X : X \in LD, X \in B^n\} \\ & LD = \{X : AX \leq b, X \geq 0\} \\ \text{avec} & A = (m \times n), c^k(1 \times n), X(n \times 1), b(m \times 1) \\ \text{et} & B = \{0, 1\} \end{array}\right]$$

Dû aux caractères contradictoires des objectifs, il n'existe pas de solution minimisant simultanément chacun des objectifs (et c'est pour cette raison que l'on utilise la notation $\ll \min \gg$), mais un ensemble de solutions dites *efficaces*. Une solution $X^* \in D$ est efficace pour le problème (P) s'il n'existe pas d'autres solutions $X \in D$ telles que : $z_k(X) \leq z_k(X^*), k = 1, \ldots, K$ avec au moins une inégalité stricte. Nous désignerons par $E(P)$ l'ensemble des solutions efficaces du problème (P), que l'on appelle également *front Pareto* ou *surface de compromis*.

Il n'est plus à montrer l'efficacité des métaheuristiques pour la résolution des problèmes d'optimisation combinatoire. Elles ont été récemment adaptées aux problèmes multiobjectifs [1,3,5], et constituent un domaine de recherche en plein développement.

Cet article est organisé de la façon suivante : nous présentons dans les deux premières sections les deux métaheuristiques développées : les méthodes `PRTS+D` et `MEMOTS`, qui sont des méthodes générales de résolution des problèmes d'optimisation combinatoire multicritère. Nous exposons ensuite comment ces méthodes ont été adaptées au problème du sac à dos multidimensionnel multiobjectif. Nous comparons finalement leurs performances vis-à-vis de différentes métaheuristiques multicritères.

2 Présentation de `PRTS+D`

La méthode tabou `PRTS+D` (`P`areto `R`anking `T`abu `S`earch + `D`ensity) est une méthode suivant le schéma classique de la métaheuristique tabou [6], mais s'appuyant sur une fonction d'évaluation originale des voisins.

La fonction d'évaluation des voisins est basée sur un article récent de Elaoud et al. [4]. Ils y présentent un algorithme génétique (dénommé `PFGA` pour `P`areto `F`itness `G`enetic `A`lgorithm) reposant sur une nouvelle fonction d'évaluation des individus basée sur leur rang et leur densité. Cette approche donnant des résultats prometteurs, la fonction d'évaluation de `PFGA` a été adaptée dans le cadre d'une recherche tabou pour l'évaluation et la sélection des voisins.

Deux mesures interviennent dans la fonction d'évaluation de `PFGA` : un classement des individus selon un rang original et une division de l'espace des objectifs en hypervolumes permettant de mesurer la densité des individus. Ces deux notions sont ensuite agrégées dans une fonction d'évaluation unique.

Il est à préciser que nous allons parler ici d'individus et de population. Nous définirons par la suite les individus et la population à prendre en compte dans le cas de l'algorithme tabou `PRTS+D`.

2.1 Le double rang de Pareto *DPR*

Le rang *DPR* (*D*ouble *P*areto *R*anking) [4] d'un individu i est déterminé en deux temps. Dans un premier temps, on compte le nombre d'individus de la population qui dominent l'individu i, ce qui revient au classique rang de Pareto :

$$PR(i) = \Big|\{j | j \in P, i \prec j\}\Big|,$$

où P représente la population d'individus et le symbole $\prec$ correspond à la relation de dominance forte.

On définit ensuite le rang *DPR* d'un individu i comme étant la somme de son rang Pareto $PR(i)$ et du rang Pareto $PR(j)$ de ses dominateurs j :

$$DPR(i) = PR(i) + \sum_{j \in P,\, j \succ i} PR(j)$$

La valeur du rang *DPR* sera donc toujours supérieure ou égale à la valeur du rang *PR*, et augmentera plus vite si le voisin est mal classé.

A titre d'exemple, on peut voir la valeur des rangs *PR* et *DPR* pour une population de 13 individus à la figure 1. Si on compare les individus I_1 et I_2, on constate que leurs rangs *PR* sont égaux puisqu'ils sont chacun dominés par quatre individus. L'individu I_1 sera par contre préféré étant donné que les rangs *PR* des individus qui le dominent sont plus faibles, ce qui est traduit par un rang *DPR* inférieur à celui de l'individu I_2.

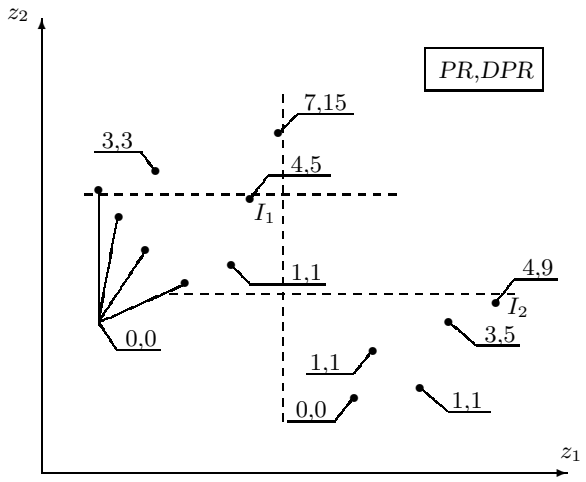

Fig. 1. Illustration du rang *DPR* utilisé dans `PFGA`.

2.2 Mesure de la densité des individus

La notion de densité est utilisée dans PFGA comme facteur de « sharing » pour maintenir une distribution équitable des individus dans l'espace de recherche et pour favoriser des zones non explorées.

Pour mesurer la densité des individus, Elaoud et al. [4] réalisent une division de l'espace des objectifs en hypervolumes et calculent le nombre d'individus présents dans chaque hypervolume. La densité $D(i)$ d'un individu i est ainsi définie par le nombre d'individus se situant dans le même hypervolume que l'individu i.

Il existe plusieurs possibilités pour la création des hypervolumes dans l'espace des objectifs. Knowles and Corne utilisent dans PAES [11] des hypervolumes de tailles égales. Elaoud et al. considèrent également des hypervolumes de tailles égales, sauf pour ceux situés le plus près de la frontière Pareto qui sont de tailles réduites. Ainsi, si un individu est proche de la frontière Pareto, il se situera dans un hypervolume de plus petite taille. Il a donc plus de chances d'avoir une faible fonction de densité, c'est-à-dire que d'autres individus se situent dans son hypervolume. Les individus présentant une faible densité seront ensuite favorisés dans la fonction d'évaluation.

Nous améliorons cette approche en utilisant dans PRTS+D des tailles d'hypervolumes variant selon une suite géométrique. Plus on se rapproche de la frontière Pareto, et plus les dimensions des hypervolumes diminuent. Pour ce faire, on va réaliser la division de chaque axe des objectifs k pour former p parties ($j = 1, \ldots, p$) de longueur $W_{k,j}$ vérifiant une suite géométrique, de manière à former p^K hypervolumes. Dans un premier temps, la longueur de la plus grande partie selon chaque objectif k est déterminée par la relation suivante :

$$W_{k,1} = \frac{\max\limits_{i \in P} z_k(X_i) - \min\limits_{i \in P} z_k(X_i)}{\left(\frac{1-\alpha^p}{1-\alpha}\right)} \quad \forall\, k \in [1, K]$$

où $\max\limits_{i \in P} z_k(X_i) - \min\limits_{i \in P} z_k(X_i)$ représente la longueur de l'axe de l'objectif k à diviser en p parties et α est tel que $0 < \alpha < 1$.

Les longueurs des parties suivantes observent une décroissance basée sur une suite géométrique de paramètre α :

$$W_{k,j} = \alpha\, W_{k,j-1} \quad \forall\, k \in [1, K],\ \forall\, j \in [2, p],\ \text{de sorte que :}$$

$$\begin{aligned} \sum_{j=1}^{p} W_{k,j} &= W_{k,1} \cdot (1 + \alpha + \ldots + \alpha^{p-1}) \\ &= \max_{i \in P} z_k(X_i) - \min_{i \in P} z_k(X_i) \end{aligned}$$

Le paramètre α compris entre zéro et un, permet de définir la rapidité de décroissance des longueurs des parties. Plus la valeur de α est petite, plus la taille des parties diminue rapidement.

On peut voir à la figure 2 les hypervolumes générés dans le cas bicritère avec $\alpha = 0.8$ et $p = 4$. Un hypervolume étant de dimension K, les hypervolumes

sont représentés dans le cas bicritère par des surfaces, et l'ensemble des surfaces forment un quadrillage à l'aide duquel on peut facilement mesurer la densité de chaque individu en comptant le nombre d'individus présent dans les surfaces.

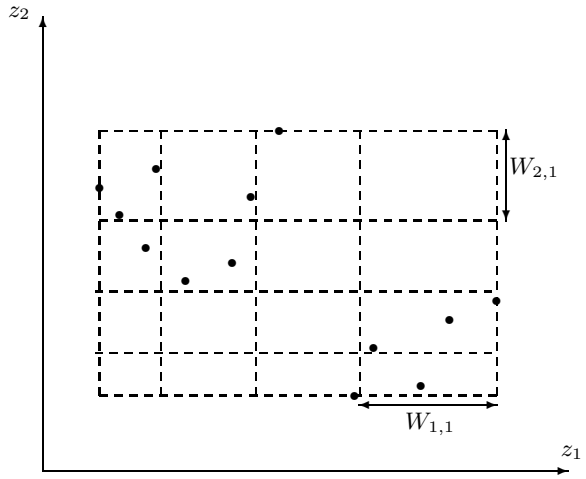

Fig. 2. Hypervolumes générés dans le cas bicritère avec $\alpha = 0.8$ et $p = 4$.

2.3 Fonction d'agrégation de *DPR* et de la densité

En agrégeant la notion de double rang de Pareto et la fonction de densité, on obtient une nouvelle fonction d'évaluation d'un individu i. La fonction d'agrégation proposée par Elaoud et al. [4] est la suivante :

$$f(i) = \frac{1}{e^{DPR(i)} \times D(i)}$$

Un individu sera d'autant mieux évalué qu'il présente une fonction d'évaluation proche de l'unité. Dans le cas où l'individu n'est pas dominé, son rang *DPR* est nul et si aucun individu ne se situe dans sa zone, sa densité est de 1. Sa fonction d'évaluation aura donc une valeur unitaire, et c'est le maximum que l'on peut atteindre. Une importance exponentielle est donnée au rang, car le rang *DPR* inclut déjà une information sur la densité. La fonction de densité D apporte une information supplémentaire sur la distribution de la population dans l'espace des objectifs, et permet d'arbitrer les égalités entre les individus présentant une valeur égale pour le rang *DPR*.

2.4 Adaptation de *DPR* et de la fonction de densité *D* à la méthode tabou

Nous utilisons la fonction d'agrégation du rang *DPR* et de la densité D pour le choix du meilleur voisin de la solution courante. Les individus pour lesquels on

évalue la fonction d'agrégation seront donc les voisins. La population comprendra les L voisins générés à chaque itération ainsi que les solutions potentiellement efficaces trouvées jusque là par l'algorithme. Les hypervolumes sont créés à chaque itération en fonction de la population courante.

A chaque itération, on évalue chaque voisin et le voisin choisi sera celui qui présente une évaluation maximale pour la fonction d'agrégation du rang DPR et de la densité définie à la section 2.3. On actualise ensuite la liste des solutions potentiellement efficaces PE par ajout des voisins présentant une valeur nulle pour le rang DPR et en éliminant les solutions potentiellement efficaces qui pourraient se retrouver dominées.

L'algorithme de la méthode `PRTS+D` est décrit à la figure 3.

Paramètres

m : taille de la liste tabou
L : nombre de solutions voisines engendrées à chaque itération (longueur du voisinage)
n : nombre d'itérations
p : nombre de parties créées le long des axes des objectifs
α : paramètre de la suite géométrique définissant la longueur des parties

Notations

$PE \equiv$ liste des solutions non dominées, potentiellement efficaces
$T \equiv$ liste tabou de taille m
$X \equiv$ solution courante
$V(X) \equiv$ voisinage de X : ensemble des solutions $\in D$ obtenues par un mouvement à partir de X
$DPR(Y) \equiv$ double rang de Pareto d'un voisin Y
$D(Y) \equiv$ densité d'un voisin Y

Initialisation

Choisir aléatoirement une solution $X_0 \in D$
$PE \leftarrow \{X_0\}$
$X \leftarrow X_0$
$T \leftarrow \emptyset$

Itération i

Générer aléatoirement L solutions voisines $Y_1, \ldots, Y_l, \ldots, Y_L$ non tabou dans $V(X)$
Créer p^K hypervolumes dans l'espace des objectifs
`Pour` chaque voisin Y_l calculer $f(Y_l) = \frac{1}{e^{DPR(Y_l)} \times D(Y_l)}$
Calculer $f^* = f({Y_p}^*) = \max_{l=1,\ldots,L} f(Y_l) \quad p = 1, \ldots, P$ ($P > 1$ en cas d'ex æquo)
Choisir aléatoirement un voisin Y^* parmi $\{{Y_p}^*, p = 1, \ldots, P\}$
$X \leftarrow Y^*$
Donner au mouvement $X \rightarrow Y^*$ le statut tabou
`Pour` chaque voisin Y_l, `Si` $DPR(Y_l) = 0$ `Alors`
  $PE \leftarrow PE + \{Y_l\}$
  Actualiser PE

Critère d'arrêt

Nombre d'itérations $i = n$

Fig. 3. Algorithme tabou `PRTS+D`.

3 Présentation de MEMOTS

Dans le but d'améliorer les performances de PRTS+D, nous nous sommes inspirés des techniques utilisées par MOGLS [10], où une bonne solution issue d'une population est améliorée par une méthode de recherche locale. La méthode MEMOTS ainsi développée est un algorithme mémétique [12], c'est-à-dire hybridant un algorithme génétique et une méthode de recherche locale. La méthode de recherche locale utilisée dans MEMOTS n'est pas exactement la méthode PRTS+D présentée à la section 2. En effet, dans un algorithme mémétique, le rôle de la recherche locale est d'intensifier la recherche. Ainsi, nous n'utilisons pas la densité D dans la fonction d'évaluation de PRTS+D, mais uniquement le double Pareto ranking, ce qui amène à la méthode PRTS.

Le fonctionnement de l'algorithme est le suivant. Dans une phase d'initialisation, la méthode PRTS est appliquée jusqu'à un certain nombre it_{stop} d'itérations sans amélioration. Il est facile de mesurer une amélioration dans la surface de compromis dans le cas de PRTS, puisqu'il y a amélioration lorsque le rang DPR d'un voisin est nul.

Les solutions potentiellement efficaces trouvées lors de cette phase d'initialisation vont représenter la population initiale. Ensuite, comme dans MOGLS, deux solutions de la population vont être sélectionnées. La première solution choisie est une des solutions de la population présentant une densité minimale, calculée à partir de la division en hypervolumes présentée à la section 2.2, la seconde est une des solutions les plus proches de la première solution au sens de la distance euclidienne dans l'espace des critères. Ces deux solutions sont ensuite combinées grâce à un opérateur de croisement pour ainsi obtenir une nouvelle solution. On applique enfin l'algorithme PRTS à partir de cette solution jusqu'à satisfaction du même critère d'arrêt qu'utilisé dans la phase d'initialisation. L'algorithme de la méthode MEMOTS est donné à la figure 4.

4 Application au problème du sac à dos multidimensionnel multiobjectif

Le problème du sac à dos consiste, dans sa version unidimensionnelle et unicritère, à choisir un ensemble d'objets à déposer dans un sac. Chaque objet ayant un poids et un certain profit, il faut sélectionner les objets maximisant le profit total tout en ne dépassant pas la charge admissible du sac. Dans le cas multiobjectif, à chaque objet est associé K profits et dans le cas multidimensionnel, q contraintes sont adjointes au sac, qui conduit à la définition suivante :

$$\begin{bmatrix} \max z^k(X) = \sum_{i=1}^{n} c_i^k x_i & k = 1, \ldots, K \\ \text{t.q.} \quad \sum_{i=1}^{n} w_i^l x_i \leq b_l & l = 1, \ldots, q \\ x_i \in \{0, 1\} & i = 1, \ldots, n \end{bmatrix}$$

Avec n le nombre d'objets disponibles, X le vecteur des choix x_i ($x_i = 1$ signifie que l'objet i est dans le sac), c_i^k le profit k de l'objet i, w_i^l le poids l de l'objet i, b_l la valeur de la charge admissible l et z^k la valeur prise par l'objectif k.

Paramètres

m, L, α, p : paramètres de la méthode `PRTS`
it_{stop} : nombre d'itérations sans amélioration de la méthode `PRTS`
r : nombre de solutions prises en compte pour la sélection de la solution la plus proche
n : nombre d'itérations

Notations

$PE \equiv$ liste des solutions non dominées, potentiellement efficaces
$T \equiv$ liste tabou de taille m
$D(X) \equiv$ densité d'une solution potentiellement efficace X

Initialisation

Choisir aléatoirement une solution $X_0 \in D$
$PE \leftarrow \{X_0\}$
$T \leftarrow \emptyset$
Appliquer `PRTS`, arrêt si plus d'amélioration pendant it_{stop}

Itération i

`Pour` chaque solution de PE calculer $D(X_l) \;\; l = 1, \ldots, |PE|$
Calculer $D^* = D({X_p}^*) = \min_{l=1,\ldots,|PE|} D(X_l)$ $p = 1, \ldots, P$ ($P > 1$ en cas d'ex æquo)
Choisir aléatoirement une solution X_1 parmi $\{{X_p}^*, p = 1, \ldots, P\}$
Choisir une solution X_2 parmi les r solutions de PE les plus proches de X_1
Croiser les deux solutions X_1 et X_2 pour ainsi obtenir X_3
$PE \leftarrow PE \; + \{X_3\}$
Actualiser PE
$T \leftarrow \emptyset$
Appliquer `PRTS` en partant de X_3, arrêt si plus d'amélioration pendant it_{stop}

Critère d'arrêt

Nombre d'itérations $i = n$

Fig. 4. Algorithme mémétique `MEMOTS`.

4.1 Adaptation de `PRTS+D`

Les deux principaux éléments à définir pour adapter l'algorithme `PRTS+D` à la résolution du problème du sac à dos sont le voisinage et la gestion de la liste tabou.

Voisinage de `PRTS+D` Le voisinage $V(X)$ utilisé est défini de la manière suivante :

- on fait sortir du sac un objet i ($x_i = 1$) choisi aléatoirement.
- on remplit ensuite au maximum le sac avec les objets j non présents dans le sac ($x_j = 0$) pris dans l'ordre décroissant des rapports suivants : $\frac{\sum_{k=1}^{K} \lambda_k c_j^k}{\sum_{l=1}^{q} w_j^l}$.

Les poids λ_k utilisés pour calculer les rapports sont choisis aléatoirement pour chaque voisin. Le mouvement peut ainsi être caractérisé par l'entier i représentant l'objet sorti du sac et par le vecteur de poids utilisé pour sélectionner les objets rentrants dans le sac. Ces deux variables sont considérées comme l'attribut du mouvement.

Gestion de la liste tabou A chaque mouvement $X \leftarrow Y^*$, nous enregistrons une partie seulement de l'attribut du mouvement, à savoir l'indice de l'objet qui est sorti du sac. Nous interdisons ensuite le choix de cet objet pour remplir le sac pour les m (taille de la liste tabou) prochaines itérations.

4.2 Adaptation de MEMOTS

Nous devons définir l'opérateur de croisement et une procédure de réparation pour une solution ne respectant pas les contraintes du problème du sac à dos, ce qui peut arriver après croisement des deux solutions X_1 et X_2.

Le croisement est réalisé grâce à un opérateur à un point à partir des deux solutions X_1 et X_2. Comme cette opération peut générer deux solutions, nous ne gardons que celle qui possède la partie de gauche de X_1 et la partie de droite de X_2, notée X_3.

X_3 peut ensuite ne pas respecter les contraintes du problème du sac à dos. Nous appliquons donc une procédure de réparation, qui consiste à enlever les objets dans l'ordre croissant des rapports $\frac{\sum_{k=1}^{K} \lambda_k c_j^k}{\sum_{l=1}^{q} w_j^l}$ jusqu'à satisfaction des contraintes. Dans MOGLS, le jeu de poids utilisé est celui qui a permis de construire la fonction d'utilité utilisée pour la sélection de X_1 et X_2. La méthode MEMOTS n'utilise pas de fonction d'utilité, et l'emploi d'un jeu de poids aléatoire pourrait aboutir à une solution X_3 se situant assez loin de X_1 et X_2, et n'étant donc plus située dans une zone de densité minimale. Nous allons donc déterminer le jeu de poids en fonction de la solution efficace X_1 choisie, de façon à garantir que plus l'évaluation de la solution X_1 est bonne selon un critère vis-à-vis des solutions potentiellement efficaces trouvées jusque là, plus la valeur du poids selon ce critère sera grand. La formule de calcul du jeu de poids λ est ainsi la suivante :

$$\lambda_k = \frac{R\big(k, z_k(X_1)\big)}{\sum_{k=1}^{K} R\big(k, z_k(X_1)\big)} \qquad k = 1, \ldots, K$$

où la fonction $R\big(k, z_k(X_1)\big)$ donne le nombre de solutions potentiellement efficaces dont l'évaluation selon le critère k est inférieure à $z_k(X_1)$. La valeur de cette fonction évolue entre 0 et $|PE| - 1$, où $|PE|$ représente le nombre de solutions potentiellement efficaces.

De cette façon, nous obtenons un jeu de poids pour la procédure de réparation qui va dans la direction de la solution de densité minimale X_1.

4.3 Jeux de données

Nous utilisons trois instances du sac à dos publiées par Zitzler [15]. Les jeux d'essai comportent 250, 500 et 750 objets, deux objectifs et deux contraintes.

Les solutions efficaces des instances à 250 et 500 objets sont disponibles [15], ce qui est particulièrement utile pour mesurer la qualité d'une approximation. Pour l'instance à 750 objets, un ensemble de solutions de référence a été obtenu en exécutant plusieurs fois les méthodes MEMOTS et MOGLS durant un grand

nombre d'itérations. Nous retenons ensuite uniquement les solutions non dominées, auxquelles une certaine valeur ϵ est ajoutée.

5 Indicateurs de qualité

Dans le cas unicritère, il est aisé de mesurer la qualité d'une solution ou de comparer les solutions obtenues par différentes méthodes. Cela est beaucoup moins simple dans le cas multicritère [16], car les solutions sont représentées par une surface de compromis composée d'un ensemble de solutions.

Nous utilisons donc plusieurs indicateurs pour la comparaison des performances des algorithmes :

- l'hypervolume $\mathcal{H}$ (à maximiser) [15] : approximation du volume compris sous la courbe formée par les points de l'ensemble à évaluer.
- la métrique d'espacement ME (à minimiser) [13] : mesure l'uniformité de la répartition des solutions composant la surface de compromis. Plus la valeur est proche de zéro, meilleure sera la répartition des solutions sur la surface de compromis.
- la mesure R (à minimiser) corrélée au volume compris entre un point de référence et les solutions de l'approximation. Les points de référence utilisés peuvent être trouvés dans [9].
- la distance moyenne D_1 entre les solutions de l'approximation et les solutions de référence (à minimiser).
- la distance maximale D_2 entre les solutions de l'approximation et les solutions de référence (à minimiser).
- le nombre de solutions potentiellement efficaces trouvées, noté $|PE|$.

Etant donné le caractère stochastique des algorithmes, nous calculons à chaque fois la moyenne des indicateurs sur vingt exécutions.

6 Paramètres des algorithmes `PRTS+D` et `MEMOTS`

Le nombre p de parties utilisé par `PRTS+D` et `MEMOTS` est fixé dynamiquement par la relation suivante, arrondie à la dizaine supérieure :

$$p = \sqrt[\kappa]{N}$$

Où N représente le nombre de solutions prises en compte, c'est-à-dire les solutions potentiellement efficaces et les voisins pour `PRTS+D` et uniquement les solutions potentiellement efficaces pour `MEMOTS`. Le paramètre α qui définit la suite géométrique que suivent les longueurs des parties a été fixé à 0.7 pour toutes les instances. On peut voir à la figure 5 l'influence de α sur `MEMOTS` pour l'instance à 500 objets, mesurée par les indicateurs D_1 et D_2 qui sont considérés comme les indicateurs les plus puissants ; D_1 pour mesurer l'intensification et D_2 pour mesurer la diversification. On remarque que l'utilisation d'une suite géométrique pour la longueur des parties est tout à fait justifiée, puisque les

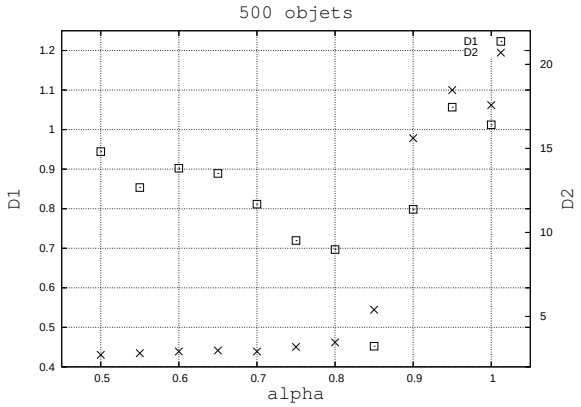

Fig. 5. Inluence de α sur MEMOTS.

performances se dégradent quand α se rapproche de l'unité (ce qui correspond à des longueurs de parties égales).

Les autres paramètres de PRTS+D et MEMOTS sont donnés au tableau 1. On peut remarquer que la taille m de la liste tabou de MEMOTS est plus petite que celle de PRTS+D, car dans MEMOTS, la recherche tabou a pour rôle d'intensifier la recherche. Pour MEMOTS, le nombre n d'itérations représente non pas le nombre de lancements de l'algorithme PRTS, mais la somme du nombre d'itérations effectuées dans chaque exécution de PRTS. Ainsi, quel que soit le nombre d'itérations de la recherche tabou effectué, le nombre d'itérations total de MEMOTS reste constant. Ce nombre a été fixé au nombre d'itérations utilisé par Jaszkiewicz dans MOGLS.

Tab. 1. Paramètres de PRTS+D et MEMOTS.

Instance	n	L	m
250	20000	100	17
500	25000	200	34
750	30000	300	51

Instance	n	L	m	r	it_{stop}
250	7500	100	3	20	3
500	10000	200	5	20	7
750	12500	300	7	20	10

Le nombre it_{stop} d'itérations sans amélioration de PRTS a un rôle important dans MEMOTS, comme on peut le voir à la figure 6 où nous avons représenté pour l'instance à 500 objets l'évolution des indicateurs D_1 et D_2 en fonction de it_{stop}. On remarque qu'il existe une valeur optimale différente pour it_{stop} suivant l'indicateur que l'on désire favoriser. Pour optimiser D_2, c'est-à-dire la diversification, il est préférable d'arrêter assez rapidement PRTS. Par contre, pour obtenir des solutions proches du front Pareto (meilleure intensification), le nombre it_{stop} doit être plus important. Un compromis doit donc être considéré.

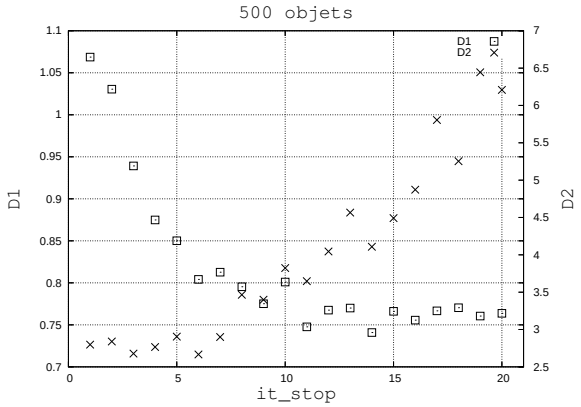

Fig. 6. Inluence de it_{stop} sur MEMOTS.

7 Résultats numériques

On peut voir au tableau 3 la comparaison des indicateurs de performances de `PRTS+D` et `MEMOTS` avec les algorithmes mémétiques `MOGLS` [10] et `IMMOGLS` [7], ainsi qu'avec l'algorithme évolutionnaire `NSGA-II` [2]. L'implémentation et les paramètres de ces méthodes ont été fixés conformément à la librairie de Jaszkiewicz [8] permettant d'évaluer plusieurs métaheuristiques multicritères.

On constate que les performances de `MEMOTS` sont supérieures à celles de `PRTS+D` et ce d'autant plus que le nombre d'objets augmente. L'écart entre `PRTS+D` et `MEMOTS` est particulièrement élevé pour l'indicateur D_2 qui caractérise la capacité de l'algorithme à trouver des solutions situées dans les zones extrêmes de la surface de compromis. `MEMOTS` présente également de meilleurs résultats par rapport aux autres méthodes pour pratiquement tous les indicateurs. Nous avons représenté à la figure 7 les solutions de `MEMOTS` en comparaison des solutions efficaces pour l'instance à 500 objets. On constate que les solutions sont très proches des solutions efficaces, ce qu'on pouvait également constater via le tableau des indicateurs.

Les temps d'exécution de `MEMOTS` sont donnés au tableau 2 (sur un Pentium IV cadencé à 2.4 GHz) pour les trois instances.

Tab. 2. Temps d'exécution de l'algorithme `MEMOTS`.

Instance	Temps(S)
250	9.00
500	38.91
750	102.41

Tab. 3. Comparaison des indicateurs de `PRTS+D`, `MEMOTS`, `MOGLS`, `IMMOGLS` et `NSGA-II` pour les instances à 250, 500 et 750 objets.

Instance	Méthode	$\mathcal{H}(10^7)$	ME	R	D_1	D_2	$\|PE\|$
250 objets	*Référence*	*9.87*	*2.57*	*245.60*	*0.00*	*0.00*	*568.00*
	PRTS+D	9.71	3.81	269.42	1.34	8.76	198.85
	MEMOTS	**9.86**	**3.61**	**254.92**	**0.48**	3.05	**265.95**
	MOGLS	9.82	4.39	259.44	0.86	4.42	170.70
	IMMOGLS	9.83	6.91	279.17	1.92	**2.99**	80.90
	NSGA-II	9.44	5.10	297.03	3.05	22.72	74.00
500 objets	*Référence*	*40.79*	*2.12*	*430.62*	*0.00*	*0.00*	*1397.00*
	PRTS+D	40.20	3.58	0.00	1.76	10.90	377.10
	MEMOTS	**40.74**	**3.38**	**449.24**	**0.81**	**2.90**	**488.30**
	MOGLS	40.56	4.68	463.39	1.68	4.13	250.50
	IMMOGLS	40.59	7.90	504.95	3.55	5.10	93.45
	NSGA-II	38.33	7.40	627.71	6.62	31.59	49.80
750 objets	*Référence*	*89.73*	*3.32*	*728.66*	*0.00*	*0.00*	*863.00*
	PRTS+D	87.69	3.49	804.93	3.06	13.43	633.25
	MEMOTS	**89.18**	**3.22**	**763.26**	**1.33**	**3.22**	**884.90**
	MOGLS	88.54	5.02	799.80	2.93	4.95	359.10
	IMMOGLS	88.95	9.80	862.21	4.87	7.88	98.25
	NSGA-II	81.07	9.95	1296.36	15.52	42.49	37.60

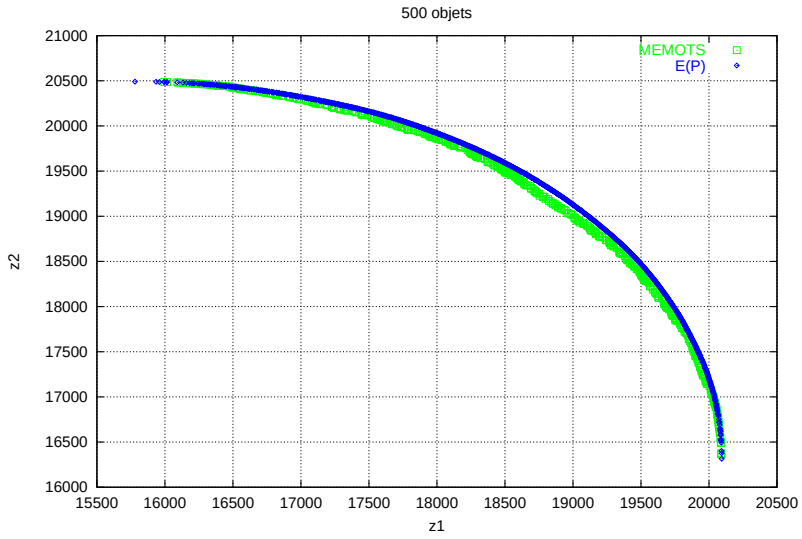

Fig. 7. Comparaison des solutions de `MEMOTS` avec les solutions efficaces $E(P)$ pour l'instance à 500 objets.

8 Conclusion

Nous avons présenté dans cet article deux nouvelles métaheuristiques pour la résolution de problèmes d'optimisation combinatoire multicritère : la méthode PRTS+D, basée sur la recherche tabou et la méthode MEMOTS, fondée sur un algorithme mémétique. Ces deux méthodes n'utilisent pas de fonction d'agrégation basée sur des jeux de poids, ce qui représente un avantage face à certaines méthodes [7,10,14], puisque l'utilisation d'une fonction d'agrégation demande une gestion efficace et pas toujours évidente des jeux de poids, mais aussi une normalisation des valeurs prises par les critères. De plus, les méthodes basées sur des fonctions d'agrégation demandent en général un nombre d'évaluations plus élevé, étant donné que des solutions mauvaises selon la fonction d'agrégation courante peuvent être rejetées même si ce sont des solutions non dominées.

Il s'est révélé que la méthode MEMOTS est plus performante que PRTS+D et que d'autres métaheuristiques multicritères. Dans PRTS+D, le choix de l'espace où est réalisée l'intensification est géré de manière complètement autonome par l'algorithme. Les moins bons résultats de PRTS+D sur les instances de grande taille ont montré que l'intensification de la méthode n'était pas assez bien guidée. L'emploi d'une population de solutions utilisée dans MEMOTS a permis de mieux diriger la recherche, grâce une sélection supervisée et diversifiée des zones d'intensification, caractéristique essentielle d'une bonne métaheuristique multicritère. Nous avons montré dans cet article qu'il n'était pas indispensable d'utiliser plusieurs fonctions d'agrégation basées sur des jeux de poids pour obtenir cette propriété. Toutefois, pour mieux attester des bonnes performances de MEMOTS, il serait nécessaire d'appliquer la méthode sur d'autres problèmes d'optimisation combinatoire multicritère, comme le problème du voyageur de commerce ou le problème d'affectation. Il serait également intéressant de tester la méthode sur des instances à plus de deux critères et d'étudier l'influence du nombre d'itérations de la méthode tabou sur les performances de MEMOTS pour ces instances où la méthode d'intensification a une importance moindre, étant donné qu'il est plus aisé de trouver une solution non dominée vu le nombre de solutions plus élevé.

Références

1. Collette, Y., and Siarry, P. *Optimisation multiobjectif.* Eyrolles, 2002.
2. Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization : NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.
3. Ehrgott, M., and Gandibleux, X. *Multiple Criteria Optimization : State of the Art Annotated Bibliographic Surveys.* Kluwer Academic Publishers, Boston, 2002.
4. Elaoud, S., Loukil, T., and Teghem, J. Pareto fitness genetic algorithm. A paraître dans European Journal of Operational Research, 2005.
5. Gandibleux, X., Sevaux, M., Sörensen, K., and T'Kindt, V. *Metaheuristics for Multiobjective Optimisation.* Springer, 2004.

6. Glover, F., and Laguna, M. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
7. Ishibuchi, H., and Murata, T. Multi-Objective Genetic Local Search Algorithm. In *Proceedings of the 1996 International Conference on Evolutionary Computation* (Nagoya, Japan, 1996), T. Fukuda and T. Furuhashi, Eds., IEEE, pp. 119–124.
8. Jaszkiewicz, A. Experiments done with the momhlib : http ://www-idss.cs.put.poznan.pl/ jaszkiewicz/MOMHLIB/. Tech. rep., Institute of Computing Science, Poznan University of Technology, 2000.
9. Jaszkiewicz, A. On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment. Tech. Rep. RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznań, Poland, July 2000.
10. Jaszkiewicz, A. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research 137*, 1 (2002), 50–71.
11. Knowles, J. D., and Corne, D. W. The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation* (Washington, D.C., July 1999), IEEE Service Center, pp. 98–105.
12. Moscato, P. On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. Tech. Rep. C3P 826, Caltech Concurrent Computation Program, 1989.
13. Schott, J. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. PhD thesis, Institute of Technology, Department of Aeronautics and Astronautics, Massachusetts, 1995.
14. Ulungu, E., Teghem, J., Fortemps, P., and Tuyttens, D. MOSA Method : A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multi-Criteria Decision Analysis 8*, 4 (1999), 221–236.
15. Zitzler, E. *Evolutionary Algorithms for Multiobjective Optimization : Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
16. Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C. M., and Grunert da Fonseca, V. Why Quality Assessment of Multiobjective Optimizers Is Difficult. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)* (San Francisco, California, July 2002), W. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, Eds., Morgan Kaufmann Publishers, pp. 666–673.