Problème de tournées de véhicules avec chargement bidimensionnel et contraintes de conflits partiels multi-objectif

Khaoula Hamdi-Dhaoui, Nacima Labadie, Alice Yalaoui

9e Conférence Internationale de Modélisation, Optimisation et SIMulation - MOSIM'12
06 au 08 juin 2012 - Bordeaux - France
« Performance, interopérabilité et sécurité pour le développement durable »

Problème de tournées de véhicules avec chargement bidimensionnel et contraintes de conflits partiels multi-objectif

Khaoula HAMDI-DHAOUI, Nacima LABADIE Alice YALAOUI

LOSI-ICD-UMR-STMR-CNRS Université de Technologie de Troyes
12 rue Marie Curie
10010 Cedex Troyes
khaoula.hamdi@gmail.com, nacima.labadie@utt.fr, alice.yalaoui@utt.fr

RÉSUMÉ : *Le problème de tournées de véhicules avec chargement bidimensionnel et conflits partiels consiste à combiner deux problèmes NP-difficiles : le problème classique de tournées de véhicules et le problème de placement en deux dimensions avec conflits partiels. Nous proposons dans ce papier un algorithme de type NSGA-II pour réaliser une étude bi-objective dans laquelle nous considérons la minimisation du coût total de transport ainsi que l'équilibrage de la charge entre les différentes tournées, en termes de surface occupée.*

MOTS-CLÉS : *Tournées de véhicules, bin-packing, conflits partiels, multi-objectif.*

1 Introduction

Le problème de tournées de véhicules et de placement en deux dimensions et conflits partiels (2LPC-CVRP pour *two-dimensional Loading Capacitated Vehicle Routing Problem with Partial Conflicts*) est un couplage de deux problèmes NP-difficiles. Il consiste à minimiser le coût du transport relatif aux tournées (CVRP pour *Capacitated Vehicle Routing Problem*), tout en trouvant une solution réalisable pour le placement des demandes bidimensionnelles des clients affectés à un même véhicule et en respectant une distance de sécurité entre les objets partiellement conflictuels (2BPPC pour *Two-dimensional Bin Packing Problem with Partial Conflicts*).

La notion de contrainte de distance a été introduite dans la littérature dans différentes variantes du *bin-packing* en deux dimensions. (Stoyan & Yaskov 1998) proposent une étude dans laquelle chaque objet doit être séparé de tous les autres objets ainsi que des bordures du *bin* d'une distance donnée propre à chaque objet. Une généralisation du problème pour le cas tridimensionnel a été proposée par (Stoyan & Chugay 2009). Dans une autre variante, la distance maximale séparant deux objets affectés au même *bin* ne doit pas dépasser une valeur maximale. Ce type de problème a été étudié par (O. Beaumont & Larchevêque 2010). Le 2BPPC a été introduit par (Hamdi-Dhaoui, Labadie & Yalaoui 2011a). Les auteurs ont proposé un modèle mathématique, plusieurs heuristiques ainsi qu'un algorithme génétique à départs multiples (Hamdi-Dhaoui, Labadie & Yalaoui 2011b).

Quelques travaux récents se sont intéressés au problème de tournées de véhicules et de chargement d'objets bi-dimensionnels. Une méthode exacte a été développée par (Iori, Salazar-Gonzalez & Vigo 2007) elle utilise une méthode de *branch-and-cut* pour le routage et un *branch-and-bound* pour le placement. Des méthodes approchées ont été développées également pour la résolution du problème. La recherche taboue a été utilisée par (Gendreau, Iori, Laporte & Martello 2008) et par (Zachariadis, Tarantilis & Kiranoudis 2009). Les deux méthodes font appel à plusieurs heuristiques classiques pour le placement. Un algorithme de colonies de fourmis a été mis en oeuvre par (Fuellerer, Doerner, Hartl & Iori 2009) pour résoudre le problème. Le problème de chargement est traité à l'aide de plusieurs heuristiques classées par ordre de complexité et d'une recherche locale permet d'améliorer la qualité des solutions. Les fourmis se chargent de construire les tournées. Récemment, (Leung, Zhou, Zhang & Zheng 2011) ont proposé un algorithme de *recherche taboue étendue*. Les auteurs utilisent des techniques proches de celles développées par (Zachariadis et al. 2009) et les complètent par de nouvelles méthodes pour le problème de placement en 2D. Ils montrent qu'ils obtiennent de meilleurs résultats que (Zachariadis et al. 2009). (Duhamel, Lacomme, Quilliot & Toussaint 2011) ont proposé un GRASPxELS hybride combiné à une approche de résolution du problème de placement se basant sur sa relaxation en un problème d'ordonnancement. Ils s'intéressent au cas non séquentiel orienté et non orienté. Leur méthode permet d'améliorer les résultats pour plusieurs instances et obtient, en moyenne les

meilleurs résultats connus pour le problème.

(Khebbache, Prins, Yalaoui & Reghioui 2011) s'est intéressée au 2L-CVRP avec fenêtres de temps. L'auteur propose six heuristiques et une métaheuristique pour sa résolution. On trouve dans la littérature également des méthodes de résolution du 3L-CVRP avec fenêtres horaires (le chargement est tridimensionnel). (Moura & Oliveira 2009) ont mis en oeuvre des heuristiques simples de construction de tournées. (Moura 2008) a développé une approche multi-objectifs dans laquelle elle minimise le nombre de véhicules, le temps de transport total et le volume utilisé. Un algorithme génétique est utilisé pour la résolution du problème considéré.

Dans cet article nous étudions une extension bi-objective du 2LPC-CVRP. Deux objcetifs sont considérés : la minimisation du coût de transport et l'équilibrage de la charge des différentes tournées. Un algorithme génétique multi-objectif de type NSGA-II (pour *Nondominated Sorting Genetic Algorithm II*) est développé pour fournir des fronts Pareto approchés et un *Path Relinking* permet d'améliorer la qualité des fronts obtenus. Dans la section 2 nous présentons le nouveau problème bi-objectif. La section 3 est dédiée à l'explication des différents composants de notre algorithme NSGA-II. Dans la section 4, l'algorithme du Path Relinking est expliqué. Les résultats expérimentaux sont présentés dans la section 5.

2 Présentation du problème

Le problème regroupe deux problèmes NP-difficiles : le CVRP et le 2BPPC. Une solution du problème doit satisfaire les contraintes suivantes :
- Chaque client est servi par un seul véhicule.
- Chaque véhicule effectue une tournée partant et se terminant au dépôt en passant une seule et unique fois par chaque client qui lui est affecté.
- La somme des demandes des clients affectés à un véhicule ne doit pas dépasser sa capacité.
- Les objets de deux dimensions constituant les demandes des clients affectés à un véhicule doivent pouvoir être rangés dans le véhicule sans chevauchement ni débordement et en respectant la distance de sécurité entre les objets partiellement conflictuels. Nous supposons que les objets constituant la demande d'un même client appartiennent à la même classe : ils peuvent être de la même nature ou de natures différentes mais non conflictuelles.

Les objets partiellement conflictuels et affectés à un même véhicule doivent être séparés par une distande de sécurité D donnée. Cette contrainte implique la présence d'une certaine surface inutilisée dans chaque véhicule qui varie en fonction des dimensions des objets chargés et des relations de conflits qui existent entre eux. Quand plusieurs paires d'objets partiellement conflictuels sont affectées à un même véhicule, on peut avoir un déséquilibre important en termes de chargement entre différentes tournées.

Pour éviter une telle configuration, nous nous proposons d'équilibrer la surface occupée des véhicules. Cet équilibrage implique une répartition équitable des objets sur les véhicules en termes de surface et aussi en termes de conflits partiels. On obtient ainsi un problème bi-objectif avec les objectifs suivants :
- Minimiser le coût de transport total.
- Minimiser la différence de chargement entre la tournée la plus chargée et la tournée la moins chargée.

3 Algorithme génétique NSGA-II

En présence de plusieurs objectifs à minimiser dans un problème, le nombre de solutions optimales (appelées solutions Pareto optimales) augmente pour former un ensemble au lieu d'une solution optimale unique. Si aucun élément indicateur n'est disponible, aucune solution de cet ensemble ne peut être dite de meilleure qualité que les autres. Il est donc intéressant d'avoir autant de solutions Pareto optimales que possible. Afin de répondre à ce besoin, des algorithmes appelés Algorithmes Evolutionnaires Multiobjectifs (MOEAs pour *Multiobjective Evolutionary Algorithms*) ont été proposés.

NSGA (Srinivas & Deb 1995) a été parmi les premiers algorithmes évolutionnaires utilisés. L'algorithme NSGA-II (Deb, Pratap, Agarwal & Meyarivann 2002) est une version plus élaborée du NSGA original. Dans cette section, nous présentons les composantes de notre algorithme génétique de type NSGA-II dédié au problème bi-objectif défini dans la section 2 de ce document.

3.1 Algorithme génétique de base

3.1.1 Chromosome et évaluation

Les chromosomes sont codés comme des séquences de n clients sans délimiteurs de tournées. On obtient ainsi un tour géant qui sera découpé en tournées réalisables à l'aide d'une procédure de découpage utilisant l'opérateur *Split* de (Prins 2004) qui se base sur le calcul du plus court chemin avec l'algorithme de Bellman. La procédure de découpage appelle *Split* pour obtenir un découpage approché du chromosome initial. Les tournées ainsi construites et qui sont irréalisables pour le problème de placement sont concaténées pour obtenir un nouveau tour géant. Un découpage heuristique est alors appliqué en affectant les clients au fur et à mesure aux tournées et le placement des demandes est réalisé en parallèle.

Nous utilisons l'heuristique *Bottom-left-fill* adaptée

pour le placement. Cette heuristique consiste à placer chaque objet dans la position la plus basse possible tout en le justifiant à gauche. Les objets sont triés dans un ordre décroissant de leurs hauteurs. Cependant, quand un objet est en conflit avec au moins un objet dans la zone entourant sa position d'insertion, il est retiré de la liste et les objets suivants sont considérés. Le premier objet qui peut être placé sans laisser une distance de sécurié D est choisi. L'objet retiré est alors inséré à la fin de la liste.

3.1.2 Population initiale

Elle est constituée de N chromosomes. Le premier est obtenu par l'heuristique de (Clarke & Wright 1964) et les (N-1) suivantes sont générées aléatoirement. Les chromosomes sont découpés en tournées pour obtenir des solutions. L'heuristique de Clarke et Wright consiste à affecter un véhicule à chaque client pour ensuite fusionner les tournées de manière à réduire le coût total. Un tour géant est obtenu en concaténant les tournées.

3.1.3 Sélection et croisement

Les parents participants à un coisement sont sélectionnés par tournoi binaire. Ensuite, ils sont combinés en utilisant l'opérateur classique *OX* pour *Order Crossover* (Goldberg 1989). L'opérateur commence par choisir deux points de coupure i et j ($1 < i \leq j < n$) d'une manière aléatoire, dans le premier parent. Les clients situés dans les positions i et j sont copiés dans le chromosome enfant dans les mêmes positions. Les clients manquants sont ajoutés en parcourant le deuxième parent de gauche à droite et ils sont copiés dans leur ordre d'apparition. Chaque croisement permet d'obtenir un seul enfant.

3.1.4 Gestion de la population

Afin d'éviter la convergence prématurée de la population vers un minimum local, il est possible d'utiliser un outil de gestion de la population qui permet de choisir entre garder une solution ou la rejeter suivant un critère de distance aux solutions existantes. Les algorithmes génétiques avec gestion de la population ont été introduits par (Sörensen & Sevaux 2006). Nous considérons la distance consistant à calculer le nombre de "paires cassées" de (Martí, Laguna & Campos 2005). Soient deux chromosomes X et Y. Le client qui occupe la position i dans le chromosome X est noté X_i. La distance $D(X, Y)$ est égale au nombre de paires $\{X_i, X_{i+1}\}$ non adjacentes dans Y. La distance d'un chromosome à une population P s'exprime de la manière suivante :

$$D_P(Y) = min\{D(X, Y) : X \in P\} \quad (1)$$

Pour qu'une nouvelle solution Y soit acceptée dans la population P, sa distance à la population doit être supérieure à un seuil donné Δ. Quand Δ est égal à 0, il s'agit d'éliminer simplement les copies. La procédure de gestion commence par trier les chromosomes d'une population P dans un ordre décroissant de leur qualité, ce qui correspond à un ordre croissant des coûts. La taille de la population est constante, celà veut dire que pour accepter une nouvelle solution, il faut en éliminer une autre. Afin de conserver les meilleures solutions trouvées, la solution à éliminer est sélectionnée aléatoirement dans la pire moitié de la population. Le seuil Δ est ensuite mis à jour en fonction du taux de refus ou d'acceptation des solutions.

Dans ce qui suit, nous allons détailler les outils de gestion des fronts Pareto approchés proposés par (Deb et al. 2002).

3.2 Tri des solutions non dominées

Pour identifier les solutions appartenant au premier front non dominé, chaque solution peut être comparée avec toutes les autres solutions pour vérifier si elle est dominée par elles ou pas. Si la solution a domine la solution b, on note $a \prec b$. L'opération est recommencée N fois pour trouver toutes les solutions du premier front. Pour déterminer tous les fronts suivants, il faut reprendre la population en éliminant les solutions des fronts précédents et faire les comparaisons décrites précédemment. La complexité de ces opérations est de l'ordre de $O(N^3)$.

Algorithme 1 : Procédure de formation du premier front

```
Pour tout p ∈ P
        S_p = ∅
        n_p = 0
        Pour tout q ∈ P
                Si (p ≺ q) alors S_p = S_p ∪ {q}
                Sinon si (q ≺ p) alors n_p = n_p + 1
        Fin pour
        Si n_p = 0 alors
                p_rang = 1
                F_1 = F_1 ∪ {p}
        Fin si
Fin pour
```

Afin de réduire la complexité de cette phase d'identification des fronts, nous définissons deux entités :
- Un compteur de domination n_p, qui dénombre les solutions dominant la solution p
- L'ensemble des solutions dominées par la solution p, noté S_p.

Avec ces entités supplémentaires, le nombre de comparaisons est de l'ordre de $O(N^2)$ seulement. La procédure de construction du premier front est décrite dans l'algorithme 1. F_i désigne le front approché de rang i et P la population considérée. Le rang du front

auquel appartient la solution p est noté par p_{rang}. Une solution dont le compteur de domination n_p est à zéro appartient au premier front.

Pour chacune de ces solutions ayant $n_p = 0$, nous examinons son ensemble S_p et pour chaque solution q y figurant, nous réduisons son compteur n_q d'une unité. Si le compteur de domination d'une solution q dans cet ensemble devient égal à zéro, nous mettons cette solution dans un nouvel ensemble Q. Les membres de l'ensemble Q constituent le second front Pareto. L'opération est recommencée avec les membres de Q pour identifier les solutions du troisième front, ainsi de suite jusqu'à identification de tous les fronts (Algorithme 2).

Algorithme 2 : Procédure de formation des fronts d'ordre supérieur à 1

$i = 1$
Tant que $F_i \neq \emptyset$
    $Q = 0$
    Pour tout $p \in F_i$
        Pour tout $q \in S_p$
            $n_q = n_q - 1$
            Si $n_q = 0$ alors
                $q_{rang} = i + 1$
                $Q = Q \cup \{q\}$
            Fin si
        Fin pour
    Fin pour
    $i = i + 1$
    $F_i = Q$
Fin tant que

3.3 Préservation de la diversité

En plus de la convergence vers un front Pareto optimal, nous souhaitons obtenir avec notre algorithme génétique une bonne répartition des solutions. Une approche de comparaison de l'entassement (*crowded-comparison approach*) des solutions est utilisée.

Pour décrire cette approche, nous commençons par introduire une mesure d'estimation de la densité. Pour obtenir une estimation de la densité des solutions entourant une solution donnée de la population, nous calculons la distance moyenne entre le point représentant cette solution dans le graphe des objectifs et les deux points les plus proches des deux côtés de ce point (figure 1), et ce par rapport à chacun des deux objectifs. La moyenne des deux distance calculées donne la distance d'entassement de la solution considérée.

Le calcul de la distance d'entassement nécessite un tri de la population dans un ordre croissant de chaque objectif. Les solutions en positions limites (avec les valeurs la plus petite et la plus grande) ont une distance infinie. Toutes les autres solutions intermédiaires ont une valeur de distance égale à la valeur

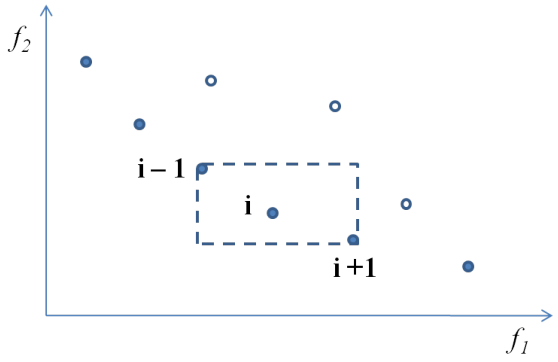

Figure 1: Mesure d'entassement (crowding)

absolue de la différence normalisée des valeurs de la fonction objectif des deux solutions adjacentes dans la liste triée (figure 1). Cette valeur est calculée pour chacun des objectifs. La distance d'entassement générale est calculée comme la somme des distances d'entassement relatives aux deux objectifs. Dans la figure 1, les cercles pleins représentent les solutions appartenant à un même front non dominé.

Dans l'algorithme 3 nous décrivons la procédure de calcul des distance d'entassement pour toutes les solutions d'un ensemble non dominé I. La procédure $trier(I, m)$ permet de trier les solutions d'une front I par ordre croissant des valeurs prises par la fonction-objectif relative à l'objectif m. $I[i].m$ désigne la valeur de la fonction-objectif relative à l'objectif m de la i^{me} solution de l'ensemble I et les paramètres f_m^{max} et f_m^{min} désignent les valeurs maximale et minimale de la fonction-objectif m.

Algorithme 3 : Procédure de calcul des distances d'entassement

$l = |I|$
Pour tout i, $I[i]_{dist} = 0$
Pour tout objectif m
    $I = trier(I, m)$
    $I[1]_{dist} = I[l]_{dist} = \infty$
    Pour $i = 2$ à $l - 1$ faire
        $I[i]_{dist} = I[i]_{dist} + (I[i+1].m - I[i-1].m)/(f_m^{max} - f_m^{min})$
    Fin pour
Fin pour

Après avoir affecté une mesure de distance à chaque solution de l'ensemble I, nous pouvons comparer deux solutions en comparant leur proximité des autres solutions. Une solution avec une valeur de la distance d'entassement plus petite est plus entassée par rapports aux autres solutions. Nous obtenons ainsi un opérateur de comparaison qui permet de déterminer un ordre partiel des solutions ($\prec_n$) en considérant deux attributs :
1- ordre de l'ensemble non-dominé (i_{rank})
2- distance d'entassement (i_{dist})

L'ordre partiel est défini comme suit :

$$(i \prec_n j) \longleftrightarrow \begin{cases} i_{rank} < j_{rank} \\ ou \\ (i_{rank} = j_{rank}) \ et \ (i_{dist} > j_{dist}) \end{cases} \quad (2)$$

Si deux solutions ont des rangs différents, la solution avec un rang inférieur est de meilleure qualité. Pour des solutions à rang égal, nous préférons la solution avec une distance d'entassement plus grande.

3.4 Structure générale

Dans cette section, nous allons décrire la boucle principale de l'algorithme génétique multi-objectif. Initialement, une population P_0 est créée en utilisant les heuristiques citées précédemment, ensuite P_0 est triée selon le critère de non-domination. Une population Q_0 de taille N est obtenue en appliquant les opérateurs de sélection, croisement et gestion décrits précédemment. Pour la population $P1$ et les populations suivantes, les étapes décrites dans l'algorithme 4 sont suivies.

Soit t le rang de la population. Tout d'abord, la population initiale P_t et la population des enfants Q_t sont fusionnées pour obtenir une population R_t de taille $2N$. La population R_t est ensuite triée selon le critère de non-domination. Les solutions du front F_1 sont bien évidemment de meilleure qualité et doivent être conservées dans la population suivante. Cependant, si la taille du front F_1 est supérieure à N seule une partie de ce front est intéressante. Dans le cas où F_1 est de taille strictement inférieure à N, la population P_{t+1} doit être complétée par des solutions des fronts suivants. Les fronts approchés sont donc déterminés successivement jusqu'à obtention d'un nombre de solutions égal à N.

Algorithme 4 : Procédure de calcul de la population t

$R_t = P_t \cup Q_t$
$F =$ Tri par non-domination de R_t
$P_{t+1} = \emptyset$ et $i = 1$
Tant que $|P_{t+1}| + |F_i| \leq N$
    Calculer la distance d'entassement de F_i
    $P_{t+1} = P_{t+1} \cup F_i$
    $i = i + 1$
Fin tant que
trier(F_i, $\prec_n$)
$FS_i =$ Les premiers $N - |P_{t+1}|$ solutions de F_i
$P_{t+1} = P_{t+1} \cup FS_i$
$Q_{t+1} =$ Enfants de P_{t+1}
$t = t + 1$

Pour obtenir exactement N individus, les solutions du dernier front F_l sont triées dans un ordre décroissant de leurs distances d'entassement en utilisant l'opérateur $\prec_n$. Comme cet opérateur nécessite les informations sur le rang du front auquel appartient la solution et sa distance d'entassement dans ce front, ces deux entités sont calculées pour chaque solution, comme expliqué dans l'algorithme 4. La méthode est également illustrée dans la figure 2, tirée de (Deb et al. 2002).

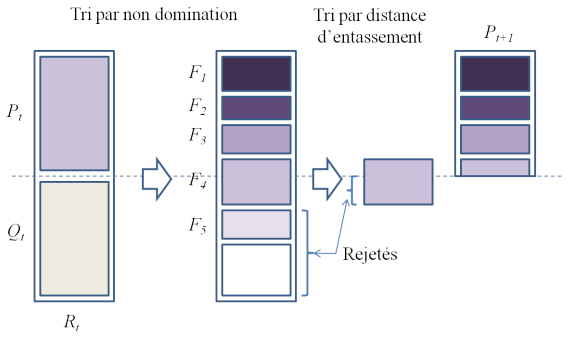

Figure 2: Mise à jour de la population

4 Path relinking

Le *Path Relinking* (PR) a été proposé par (Glover 1996) comme une stratégie d'intensification. Cette stratégie explore les trajectoires reliant deux solutions. Deux solutions sont sélectionnées dans la population ensuite en appliquant un opérateur d'exploration du voisinage, on trouve une succession de solutions intermédiaires sur la trajectoire reliant les deux solutions. La première solution est appelée *solution initiale (initiating)* S_1 et la seconde *solution cible (guiding)* S_2. Les solutions intermédiaires sont évaluées dans l'objectif de trouver de nouvelles solutions non-dominées.

Pour explorer le voisinage, nous utilisons l'opérateur *shift* qui consiste à déplacer un client de sa position sur le chromosome vers une autre position et les clients séparant les deux positions sont déplacés pour remplir la case vide. Cet opérateur a été choisi pour son côté pratique. En effet, déplacer un seul client à la fois permet d'éviter de détruire les sous-chaînes de clients construites dans les itérations précédentes. Nous utilisons une distance relative à cet opérateur qui a été proposée par (Basseur, Seynhaeve & Talbi 2005) définie comme suit : $d_{perm} = N - s_{max}$, où N est la taille du chromosome (nombre de clients) et s_{max} est la taille de la plus longue sous-chaîne commune aux solutions S_1 et S_2. Les auteurs ont prouvé que $d_{perm}(S_1, S_2)$ est une distance dans (Basseur et al. 2005). Il s'agit du nombre minimum de permutations nécessaires pour lier S_1 à S_2.

Algorithme 5 : Calcul de la plus longue sous-chaîne commune

Initialisation Pour $0 \leq i, j \leq N$: $score(i, j) = 0$
Pour $i = 1$ à N faire

Pour $j = 1$ à N faire
Si $(S_1(i) = S_2(j))$ alors
$score(i,j) = score(i-1, j-1) + 1$
$chemin(i,j) = 3$
Sinon si $(score(i-1,j) \geq score(i,j-1))$ alors
$score(i,j) = score(i-1,j)$
$chemin(i,j) = 2$
Sinon
$score(i,j) = score(i,j-1)$
$chemin(i,j) = 1$
Fin si
Fin pour
Fin pour
$d_{perm} = score(N,N)$

Calculer la plus longue sous-chaîne commune (PLCC) entre deux solutions est une application classique de la programmation dynamique (voir (Cormen, Leiserson & Rivest 1990)). Nous avons implémenté un algorithme dont la complexité est de l'ordre de $O(N^2)$ pour calculer la distance entre les solutions dans l'espace de décision. L'algorithme 5 décrit la formulation récursive permettant de calculer la longueur de la sous-chaîne commune la plus longue, alors que l'algorithme 6 permet de retrouver cette sous-chaîne.

La figure 3 donne un exemple d'application des algorithmes 5 et 6. Nous obtenons une chaîne maximale de longueur 4. La séquence (5 6 9 10) représente la plus longue chaîne commune entre les solutions S_1 et S_2. 6 mouvements sont nécessaires pour relier les deux solutions : il s'agit du nombre de clients qui ne sont pas à la même place dans S_1 et S_2. A chaque étape, un client n'appartenant pas à PLCC est déplacé de manière à le positionner entre les deux clients les plus proches dans S_2 et appartenant à PLCC. Après $d_{perm} - 1$ solutions intermédiaires générées, nous atteignons la solution S_2. Remarquons que la chaîne PLCC n'est pas unique. Les chaînes (5 7 9 10), (3 6 9 10) et (3 7 9 10) sont également des chaînes maximales communes.

Algorithme 6 : Retrouver la plus longue sous-chaîne commune

Tant que $(i > 0$ ou $j > 0)$
Si $(chemin(i,j) = 3)$ alors
$i = i - 1$
$j = j - 1$
PLCC = $S_1(i)$ + PLCC
Sinon si $(chemin(i,j) = 2)$ alors
$i = i - 1$
Sinon si $(chemin(i,j) = 1)$ alors
$j = j - 1$
Fin si
Fin tant que

Les algorithmes 5 et 6 nous ont permis de déterminer le nombre de solutions intermédiaires (égal au nombre d'opérations - 1) et la plus longue sous-chaîne commune. La dernière étape consiste à retrouver ces solutions intermédiaires pour pouvoir les évaluer. La figure 4 illustre le déroulement de l'algorithme. La

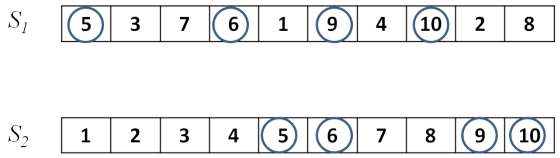

Figure 3: Calcul de la distance d_{perm}

PLCC étant (5 6 9 10), les clients 5, 6, 9 et 10 demeurent à leurs positions et les clients 1, 2, 3, 4, 7 et 8 doivent être déplacés. A la première étape, le client 3 doit être positionné avant le client 5 qui appartient à PLCC. Il existe une seule position possible pour 3 : au début du chromosome. Une fois déplacé, le client 3 devient membre de PLCC. A la deuxième étape, le client 7 doit être placé entre les clients 6 et 9 appartenant tous les deux à PLCC. Il existe deux positions d'insertions possibles : avant ou après le client 1. Quand plusieurs positions sont possibles, la position d'insertion est sélectionnée aléatoirement. Chaque client déplacé vers la position adéquate par rapport aux client de PLCC est ajouté à PLCC. A la dernière étape, la totalité des clients appartiennent à PLCC et la solution obtenue correspond à S_2.

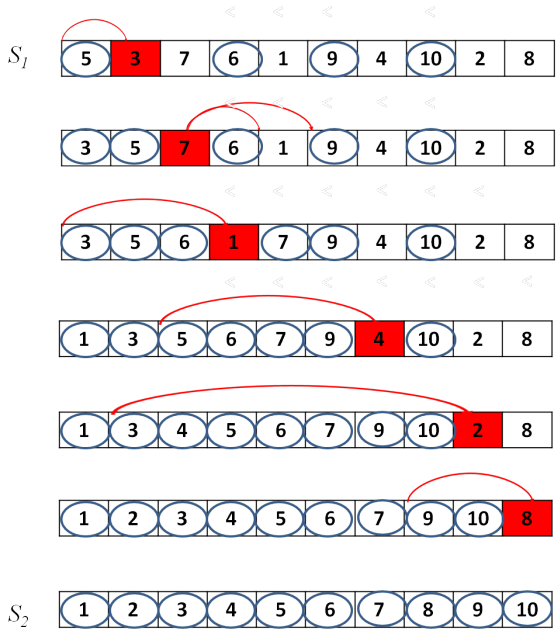

Figure 4: Exploration d'une trajectoire entre solutions dans le *Path Relinking*

5 Résultats expérimentaux

Les algorithmes sont implémentés en C++ sur un PC Pentium(R) Dual-Core CPU T4400 2.20 GHz avec un système d'exploitation Windows 7 à 64 bits. Les temps de calcul sont donnés en seconde. Dans

cette section nous allons comparer deux versions de l'algorithme : avec et sans *path relinking*. Nous allons les appeler *AGM* pour : Algorithme Génétique Multi-objectif, et *AGMPR* pour : Algorithme Génétique Multi-objectif avec Path Relinking. Nous avons testé nos deux algorithmes sur les instances de (Iori 2004) créées pour le 2L-CVRP, que nous avons adapatée à notre problème 2LPC-CVRP. Dans ces instances, le nombre d'objets et leurs dimensions ont été créés selon cinq classes. La première classe correspond à des instances d'un problème de tournées de véhicules classique (CVRP), conçues de manière à ce que le chargement n'ait aucun impact sur la construction des tournées. Dans la classe i, avec $i = 1, ..., 5$, le nombre d'objets affectés à chaque client varie entre 1 et i. Les dimensions des objets sont générés selon la loi uniforme dans des intervalles donnés (voir chapitre IV pour plus de détails). Afin d'obtenir des instances pour le 2LPC-CVRP, les objets de chaque client sont affectés à l'une des cinq cathégories d'objets partiellement conflictuelles. Un indice de classe est alors affecté à chaque client.

Nous considérons un nombre d'individus par population égal à 20 et un nombre maximums de générations évaluées égal à 100. Pour chaque instance, nous enregistrons le nombre de solutions non-dominées trouvées pour une population de 20 individus et les deux solutions extrêmes. La première solution extrême est celle qui fournit le meilleur équilibrage des charges alors que la seconde fournit le meilleur coût de transport. Afin d'évaluer la performance de l'algorithme, nous calculons la métrique de *Schott* et l'hypervolume correspondant au front des solutions non-dominées de chaque instance. La métrique de *Schott* permet de mesurer la distribution des solutions du front Pareto et elle est obtenue avec l'expression suivante :

$$\Omega = \sqrt{\frac{n}{n-1}\sum_{i=1}^{n}(\lambda_i - \overline{\lambda})^2} \quad (3)$$

Où λ_i est une distance relative au point i définie par l'equation ci-dessous et $\overline{\lambda}$ la valeur moyenne des λ_i. f_1 et f_2 sont les deux objectifs considérés :

$$\lambda_i = min_j(|f_1^i(\overrightarrow{x}) - f_1^j(\overrightarrow{x})| + |f_2^i(\overrightarrow{x}) - f_2^j(\overrightarrow{x})|) \quad (4)$$

Dans les tableaux suivants, SND désigne les Solutions Non-Dominées, $NSND$ le nombre de SND, N le nombre de clients dans l'instance considérée, $SE1$ et $SE2$ les solutions extrêmes 1 et 2, *Schott* la mesure de performance de Schott, HV la mesure de l'hypervolume et *Tps* le temps de calcul. Les objectifs considérés sont *Cout* et la différence entre la tournée la plus remplie et la moins remplie en termes de surface *Equilibre*.

Dans les tableaux 1 et 2, nous calculons la moyenne des résultats par groupe d'instances pour les deux

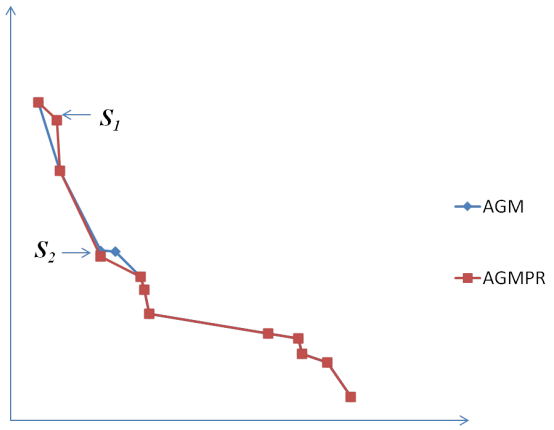

Figure 5: Fronts obtenus pour l'instance 0803

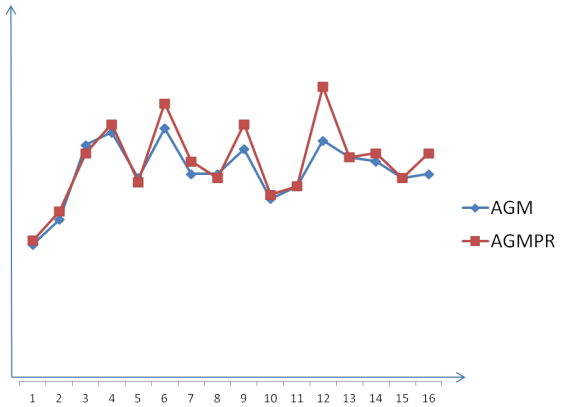

Figure 6: Variation du nombre de Solutions Non-Dominées

versions de l'algorithme. 16 groupes d'instances ont été utilisés avec un nombre de clients variant entre 15 et 35. Nous remarquons que le nombre de solutions non-dominées varie entre 6.6 et 12.2 pour AGM et entre 6.8 et 14.2 pour AGMPR. Dans la figure, 6, nous comparons les moyennes sur le nombre de SND obtenues par AGM et AGMPR. Généralement, le nombre moyen de SND trouvées par AGMPR est supérieur à celui de AGM. En moyenne, il est de10,12 pour AGM et de 10,59 pour AGMPR, soit une augmentation de 4.6%. Le nombre de SND augmente ou reste inchangé dans la quasi totalité des instances. Pour trois groupes d'instances uniquement, le nombre moyen de SND de AGMPR est légèrement inférieur à celui de AGM. Pour trois autres instances, le nombre est identique pour les deux versions.

Quand le nombre de SND est identique dans les deux versions, le front obtenu n'est pas forcément le même. En effet, certaines solutions peuvent disparaître du front quand une ou plusieurs solutions trouvées par le *path relinking* les dominent. Dans la figure 5, la

solution S_1 a été trouvée par le *path relinking* et elle est insérée entre deux solutions successives du front de AGM mais elle ne domine aucune solution de ce front, alors que la solution S_2 en domine deux. Ces deux solutions disparaissent du front de AGMPR.

La métrique de Schott a été calculée pour les fronts approchées obtenus avec les algorithmes AGM et AGMPR. La courbe obtenue dans la figure 7 indique que les solutions du front de AGMPR sont généralement mieux dispersées. Cependant, pour le groupe d'instances 13, la métrique de Schott est beaucoup plus élevée pour l'algorithme AGM. Ceci indique que les nouvelles solutions trouvées par le *path relinking* sont assez proches des solutions du front initial. En moyenne, la valeur de la métrique de Schott est de 104,13 pour AGM et de 99,68 seulement pour AGMPR, ce qui indique que les solutions sur les fronts obtenus par AGMPR sont plus proches les unes les autres. Ceci correspond effectivement au principe du *path relinking* qui consiste à trouver des solutions non-dominées entre les solutions déjà présentes sur le front non-dominé.

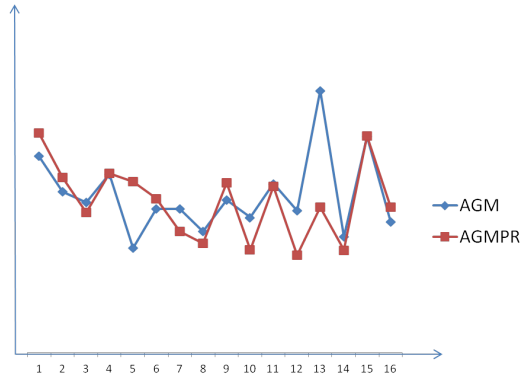

Figure 7: Variation de la métrique de Schott

La troisième mesure de performance calculée est l'hypervolume correspondant à chaque front approché. Dans la figure 8, nous expliquons la méthode de calcul employée. On considère un point de référence Z correspondant aux pires valeurs prises par les deux fonctions-objectifs durant le déroulement de tout l'algorithme. Ensuite, on calcule la surface des rectangles délimités par les verticales passant par le point de référence et la solution i et les horizontales passant par la solution i et la solution $i+1$. La somme des surfaces ainsi calculées correspond à l'hypervolume. Dans la figure 9, nous représentons la différence moyenne entre les hypervolumes des fronts de AGM et AGMPR. Généralement, la différence constatée est positive, indiquant un hypervolume plus important pour les fronts obtenus par AGMPR. Pour certains groupes d'instances, la différence est négative (groupe 9) ce qui correspond à une réduction de

Tableau 1: Résultats moyens de AGM par groupe d'instances

Inst	Nb SND	Schott	HV	Tps
1	6,6	129,69	39327,33	1,6
2	7,8	106,35	41767,23	1,6
3	11,4	99,31	45938,57	2,6
4	12	117,80	41976,74	2,4
5	9,8	69,18	60161,08	2,8
6	12,2	94,98	89901,62	3
7	10	94,98	72678,77	2,8
8	10	80,15	76806,51	3,2
9	11,2	100,80	50340,00	3,8
10	8,8	112,65	71679,22	5
11	9,4	111,49	66482,65	5,4
12	11,6	93,81	44904,37	4,8
13	10,8	172,95	203860,66	5,8
14	10,6	76,43	65730,97	7,4
15	9,8	54,13	56652,71	6,4
16	10	113,38	65649,19	6,2

la surface séparant le front obtenu par AGMPR et les droites horizontale et verticale passant par le point de référence. La valeur moyenne de l'hypervolume est de 67369 pour AGM et 69331 pour AGMPR ce qui correspond à une augmentation moyenne de 3%.

Les trois mesures de performance combinées permettent de constater l'apport positif du *path relinking*. En effet, cette procédure a permis d'améliorer la qualité des fronts approchés en augmentant le nombre de solutions non-dominées, en améliorant la dispersion des solutions sur le front et en augmentant la surface couverte par le front par rapport au point de référence.

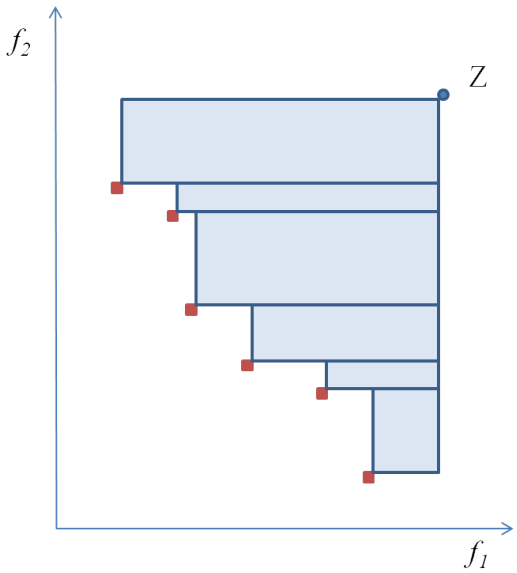

Figure 8: Méthode de calcul de l'hypervolume

Tableau 2: Résultats moyens de AGMPR par groupe d'instances

Inst	Nb SND	Schott	HV	Tps
1	6,8	144,99	40173,71	1,8
2	8,2	115,66	41778,63	1,2
3	11	92,58	45925,13	2,8
4	12,4	118,36	41986,70	2,2
5	9,6	112,74	60305,38	3
6	13,4	101,78	89932,72	3
7	10,6	79,90	78091,97	2,6
8	9,8	72,20	76960,70	3,2
9	12,4	111,98	49871,71	3,6
10	9	104,44	67805,04	5
11	9,4	109,63	66474,54	4,8
12	14,2	64,39	49110,20	4,4
13	10,8	96,18	206854,06	5,2
14	11	67,68	65743,69	6,4
15	9,8	57,04	55801,59	6,2
16	11	98,89	60367,59	6,2

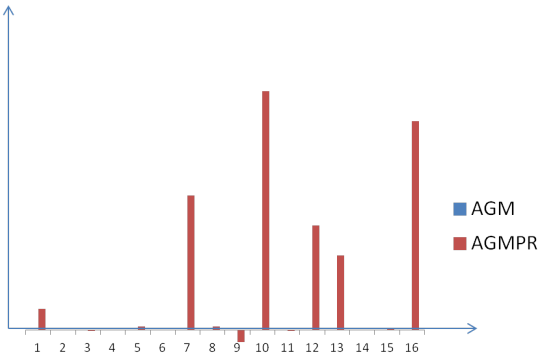

Figure 9: Variation de la différence entre les hypervolumes

6 Conclusion

Dans cet article, nous avons étudié un nouveau problème bi-objectif : le 2LPC-CVRP bi-objectif. Dans ce problème nous considérons les deux objectifs suivants : la minimisation du coût et l'équilibrage du chargement en minimisant la différence en termes de surface occupée entre la tournée la plus chargée et la tournée la moins chargée.

Pour résoudre ce problème, nous proposons un algorithme génétique de type NSGA-II, où des opérateurs de tri et de diversification de la population sont utilisés pour gérer les fronts Pareto approchés. La performance de cet algorithme est améliorée en utilisant une procédure de *path relinking*. Pour évaluer la qualité des fronts Pareto approchés obtenus, nous avons calculé le nombre de solutions non-dominées trouvées pour chaque instance ainsi que la mesure de dispersion de Schott et l'hypervolume.

Les outils cités ci-dessus nous ont permis de mesurer l'apport de la procédure de *path relinking* mais ils ne donnent pas d'indication sur la performance de la méthode en absence de solutions relatives à ce problème dans la littérature. Il est donc intéressant de développer une méthode exacte afin de pouvoir calculer la distance des fronts obtenus ici aux fronts de Pareto optimaux.

References

Basseur, M., Seynhaeve, F. & Talbi, E.-G. (2005). *Path Relinking in Pareto Multi-objective Genetic Algorithms*, Proceedings of EMO'05.

Clarke, G. & Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research* **12**: 568–581.

Cormen, T., Leiserson, C. & Rivest, R. (1990). Introduction to algorithms, *The MIT Press, Cambridge, Massachuchetts* pp. 350–355.

Deb, K., Pratap, A., Agarwal, S. & Meyarivann, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga–ii, *IEEE Transactions on Evolutionary Computation* **6**(2): 182–197.

Duhamel, C., Lacomme, P., Quilliot, A. & Toussaint, H. (2011). A multi-start evolutionary local search for the two dimensional loading capacitated vehicle routing problem, *Computers & Operations Research* **38**(3): 617–640.

Fuellerer, G., Doerner, K., Hartl, R. & Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing, *Computer & Operations Research* **36**(3): 655–673.

Gendreau, M., Iori, M., Laporte, G. & Martello, S. (2008). A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints, *Networks - Special Issue In Memory of Stefano Pallottino* **51**(1): 4–18.

Glover, F. (1996). *Tabu search and adaptive memory programing advances, applications and challenges*, Interfaces in Computer Science and Operations Research, Kluwer Academic Publishers.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachussett, USA.

Hamdi-Dhaoui, K., Labadie, N. & Yalaoui, A. (2011a). Heuristics for the bin-packing problem with partial conflicts, *12ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF' 2011), Saint-Etienne, France, 2-4 Mars* pp. 65–68.

Hamdi-Dhaoui, K., Labadie, N. & Yalaoui, A. (2011b). A memetic algorithm for the two-dimensional bin-packing problem with partial conflicts, *Proceedings of the International Conference on Industrial Engineering and Systems Management (IESM' 2011), ENIM - Metz, France, May 25-27, ISBN 978-2-9600532-3-4* pp. 371–379.

Iori, M. (2004). Metaheuristic algorithms for combinatorial optimization problems, *PhD thesis, University of Bologna, Italy* .

Iori, M., Salazar-Gonzalez, J. J. & Vigo, D. (2007). An exact approach for vehicle routing problem with two-dimensional loading constraints, *Transportation Science* **41**(2): 253–264.

Khebbache, S., Prins, C., Yalaoui, A. & Reghioui, M. (2011). Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows, *Central European Journal of Operations Research* pp. DOI: 10.1007/s10100–011–0204–9.

Leung, S., Zhou, X., Zhang, D. & Zheng, J. (2011). Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem, *Computers & Operations Research* **38**(1): 205–215.

Martí, R., Laguna, M. & Campos, V. (2005). *Scatter search vs genetic algorithms: an experimental evaluation with permutation problems*, Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search, Boston, Kluwer, C. Rego and B. Alidaee, editors.

Moura, A. (2008). *A multi-Objective Genetic Algorithm for the Vehicle Routing with Time Windows and Loading Problem*, Intelligent Decision Support, Andreas Bortfeldt.

Moura, A. & Oliveira, J. F. (2009). An integrated approach to the vehicle routing and container loading problems, *Operations Research Spectrum* **31**(4): 775–800.

O. Beaumont, N. B. & Larchevêque, H. (2010). Bin packing under distance constraint, *Technical report, Université de Bordeaux, Laboratoire Bordelais de Recherche en Informatique, INRIA Bordeaux Sud-Ouest* .

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers and Operations Research* **31**: 1985–2002.

Sörensen, K. & Sevaux, M. (2006). Ma|pm: Memetic algorithms with population management, *Computers and Operations Research* **33**(5): 1214–1225.

Srinivas, N. & Deb, K. (1995). Multiobjective function optimization usingnondominated sorting genetic algorithms, *Evolution Computation* **2**(3).

Stoyan, Y. & Chugay, A. (2009). Packing cylinders and rectangular parallelepipeds with distances between them into a given region, *European Journal of Operational Research* **360**(197): 446–455.

Stoyan, Y. & Yaskov, G. (1998). Mathematical model and solution method of optimization problem of placement of rectangles and circles taking account special constraints, *International Transactions on Operational Research* **5**(1): 45–57.

Zachariadis, E., Tarantilis, C. & Kiranoudis, C. (2009). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints, *European Journal of Operational Research* **195**(3): 729–743.