Résolution heuristique du *Stacker Crane Problem* préemptif et asymétrique à l'aide d'une *Arbre-représentation* des tournées

M. Lacroix[2*], H. Kerivin[1], A. Quilliot[2], H. Toussaint[2]

[1] Department of Mathematical Sciences, Clemson University, CLEMSON, O-326 Martin Hall, Clemson, SC 29634 - USA.
kerivin@clemson.edu

[2] LIMOS, CNRS UMR 6158, Université Blaise-Pascal - Clermont-Ferrand II, Complexe Scientifique des Cézeaux, 63177 Aubière, Cedex - France
{lacroix, quilliot, toussain}@isima.fr

Résumé : *Le Stacker Crane Problem est un problème de tournées impliquant l'utilisation d'un unique véhicule. Nous nous intéressons ici à la résolution de sa version préemptive et asymétrique. Cette résolution repose tout d'abord sur des résultats théoriques permettant de modéliser les solutions du problème sous forme d'arbre et de se ramener ainsi à un problème peu contraint. Ensuite cette représentation est utilisée pour réaliser une heuristique de construction gloutonne ainsi qu'une recherche locale simple et efficace. Enfin des résultats expérimentaux confirment l'efficacité de cette approche et l'impact de l'hypothèse de préemption.*

Mots-Clés : *Stacker Crane Problem, relais, tournée de véhicules, recherche locale, heuristique.*

1 Introduction

Les problèmes de Ramassage et Livraison (*Pickup and Delivery*), qui consistent à planifier le transport de biens ou de personnes depuis un nœud origine jusqu'à un nœud destination en utilisant un ensemble donné de véhicules, ont été largement étudiés. Plusieurs variantes ont été considérées, voir [7] et [26] pour des études et méthodes de résolution de ces problèmes. Parmi tous les problèmes de Ramassage et Livraison, le *Stacker Crane Problem* est caractérisé par le fait de disposer d'un unique véhicule, ne transportant qu'une demande à la fois. Ce problème tire son nom des grues-portiques utilisées notamment pour le chargement ou déchargement des navires. Ces grues ne peuvent manipuler qu'un conteneur à la fois, il faut donc porter une attention particulière à l'ordre des chargements et déchargements pour minimiser les déplacements de la grue.

Le *Stacker Crane Problem* (**SCP**) peut se définir de la manière suivante : étant donnés un graphe G dont les arcs sont valués par des longueurs (ou coûts) depuis un nœud "dépôt" spécifique et un ensemble de demandes K, il s'agit de déterminer la tournée d'un unique véhicule V de telle sorte que

* nouvelle adresse : lacroix@lamsade.dauphine.fr, Laboratoire LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny

chaque demande $k \in K$ soit transportée de son nœud origine o_k à son nœud destination d_k. Chaque demande k est associée à une charge unitaire et le véhicule V ne peut transporter plus d'une charge à la fois. Il s'agit donc trouver une tournée Γ dans G, la plus courte possible, qui commence et finit au dépôt et permettant à V de satisfaire toutes les demandes. Dans le **SCP** préemptif (SCPP), chaque charge peut être déchargée en n'importe quel nœud du graphe avant d'être rechargée. Ce processus peut être effectué plusieurs fois avant que la charge arrive au nœud destination. On parle de **SCP** asymétrique lorsque la fonction qui à tout arc (x, y) fait correspondre un coût $Dist(x, y)$ est asymétrique.

Le **SCP** a tout d'abord été introduit par Frederickson et al. dans [19], sous sa forme symétrique et non préemptive. Ces auteurs fournissent une preuve de sa NP-complétude en utilisant une réduction à partir du problème du Voyageur de Commerce (*Traveling Salesman Problem*, TSP). Ils fournissent également un schéma d'approximation 9/5 pour ce problème. Atallah et Kosaraju [5] ont été les premiers à traiter le cas préemptif pour le **SCP** symétrique. Ils ont étudié à la fois le cas préemptif et non préemptif dans le cas où le graphe sous-jacent est un chemin élémentaire ou un cycle élémentaire. Ils ont prouvé que dans les deux cas le problème se résout en un temps polynomial. Frederickson et Guan [17], [18] ont étudié les versions préemptives et non préemptives du **SCP** symétrique dans le cas où le graphe sous-jacent est un arbre. Plusieurs variantes du problème de Ramassage et Livraison proches du **SCP** ont été étudiées. On peut mentionner le problème du Voyageur de Commerce qui correspond au **SCP** non préemptif et dans lequel il n'y a pas de contrainte de capacité (voir [28, 14, 27]). La version asymétrique du TSP a été traitée par des approches polyhédrales, des algorithmes de coupes et branchements (*branch and cut*) [3, 4, 6, 21], et par des heuristiques [10, 20, 24]. Hernández-Pérez et González [22] se sont intéressés au TSP avec contraintes de capacité. D'autres contraintes ont été considérées relatives à l'ajout de fenêtres de temps [26], à des relations de précédence imposées par les demandes [15], [16], à une politique de chargement de type LIFO [11]. Le cas où chaque demande possède plusieurs origines et destinations, et dans lequel, toute charge ramassée à une origine de la demande peut être utilisée pour livrer n'importe quelle destination de cette même demande, a également été considérée. Cette extension, appelée le problème d'échange (*Swapping Problem*), [1, 2, 8, 9], appartient à la classe des problèmes de Ramassage et Livraison multi-origines et multi-destinations (*many-to-many Pickup and Delivery problems*), voir [7]. Le problème avec transbordements, correspondant au transport d'une demande à l'aide de plusieurs véhicules, a été étudié dans [13, 23, 25].

Nous nous intéresserons ici au *Stacker Crane Problem* Préemptif et Asymétrique, que nous noterons **SCPPA**. Nous commencerons par donner une formulation du problème, puis nous montrerons qu'il est possible de modéliser les objets solutions sous forme d'arbres. Nous verrons que cette formulation permet à la fois une résolution par programmation linéaire en nombres entiers et par des heuristiques de construction et recherches locales fournissant des résultats numériques de bonne qualité que nous présenterons en dernière partie.

2 Description formelle du SCPPA

2.1 Notations préliminaires sur les listes

Soit $\Gamma = \{x_1, \ldots, x_n\}$ une liste constituée d'objets $x_i, i \in \{1, \ldots, n\}$.

– Pour tout élément $x = x_i$ de Γ, nous noterons $Succ_\Gamma(x)$ (resp. $Pred_\Gamma(x)$), le successeur x_{i+1} (resp. prédecesseur x_{i-1}) de x dans Γ, et par $Rg_\Gamma(x)$ le rang de x dans Γ. Une tournée constituée d'un unique élément x est notée $\{x\}$, et une tournée vide est notée $\emptyset$. Le premier (resp. dernier)

élément de Γ est noté $First(\Gamma)$ (resp. $Last(\Gamma)$). Le nombre d'éléments de Γ est noté $|\Gamma|$.

– On appelle **sous-tournée** de Γ une tournée Γ' telle que $\Gamma' = \{x_{i_1}, \ldots, x_{i_p}\}$ avec $i_1 < \cdots < i_p$.

– On notera $\oplus$ l'opérateur de **concaténation** qui transforme deux séquences $\Gamma = \{x_1, \ldots, x_n\}$ et $\Gamma' = \{y_1, \ldots, y_n\}$ en une unique séquence $\Gamma \oplus \Gamma' = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$.

– On appelle **coupe** de Γ une décomposition de Γ en une concaténation $\Gamma' \oplus \Gamma''$.

– Soient x_i et x_j deux éléments de Γ tels que $i = Rg_\Gamma(x_i) \leq j = Rg_\Gamma(x_j)$. On appelle **segment** de Γ la sous-liste $\{x_i, \ldots, x_j\}$ de Γ qui est définie par tous les z tels que $i \leq Rg_\Gamma(z) \leq j$.

2.2 Modélisation du SCPPA

Nous rappelons que le **SCPPA** peut se définir de la manière suivante :

– étant donné un graphe G, un véhicule V effectue une tournée dans G afin de satisfaire un ensemble K de demandes. Chaque demande $k \in K$ est exprimée à l'aide d'un couple (o_k, d_k) de nœuds de G, tel que o_k est l'origine de la demande k, d_k est la destination de la demande k, et V doit transporter exactement une unité de charge de o_k vers d_k ;

– la capacité de chargement de V est égale à 1 ;

– le véhicule V est autorisé à traiter une demande de manière préemptive : lors du transport d'une charge C_1 il peut s'arrêter à un noeud x quelconque du réseau, décharger C_1, traiter une autre demande, puis revenir en x recharger C_1. Un tel nœud x est appelé **relais** pour la demande k ;

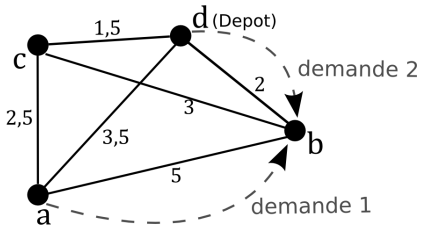
– V doit commencer et finir sa tournée à un dépôt, le nœud associé est noté *Depot*. La tournée doit être la plus petite possible au sens de la fonction coût *Dist* définie dans la partie 1.

Notons X l'ensemble des nœuds du graphe. Les nœuds origine et destination d'une demande pouvant être utilisés comme relais, on duplique ces noeuds afin de les rendre tous différents suivant leur fonction (dépôt, origine, destination ou relais). On peut ainsi décomposer X de la manière suivante : $X = \{Depot\} \cup X_O \cup X_D \cup X_R$ où :
$X_O = \{o_k, k \in K\}$;
$X_D = \{d_k, k \in K\}$;
X_R contient une copie de *Depot*, X_O et X_D et éventuellement un ensemble de relais.

Avec un calcul des plus courts chemins, on fait en sorte que ce graphe soit complet et sa description se ramène à une matrice de distance *Dist*. On suppose que cette fonction coût, non symétrique, *Dist* satisfait les inégalités triangulaires et qu'elle est telle que $Dist(x, x') = 0$ si x' est une copie de x.

La figure 1 (a) illustre un graphe à quatre nœuds "physiques" $\{a, b, c, d\}$ et deux "demandes". On notera que les arcs du graphe sont en traits continus tandis que les demandes sont symbolisées à l'aide de flèches en pointillés. Ainsi le nœud origine (o_1) de la demande 1 est situé en a, et le nœud destination (d_1) en b. Quant à la demande 2, son nœud origine (o_2) est situé en d et son nœud destination (d_2) en b. La matrice à droite de ce graphe donne pour chaque couple (x, y) de nœuds "logiques" le coût $Dist(x, y)$. Elle illustre également la décomposition de X en $X =$

$\{Depot\} \cup X_O \cup X_D \cup X_R$.



(a)

		X_O		X_D		X_R			
	Depot	o_1	o_2	d_1	d_2	a	b	c	d
Depot	0	3.5	0	2	2	3.5	2	1.5	0
o_1	3.5	0	3.5	5	5	0	5	2.5	3.5
o_2	0	3.5	0	2	2	3.5	2	1.5	0
d_1	2	5	2	0	0	5	0	3	2
d_2	0	3.5	0	2	2	3.5	2	1.5	0
a	3.5	0	3.5	5	5	0	5	2.5	3.5
b	2	5	2	0	0	5	0	3	2
c	1.5	2.5	1.5	3	3	2.5	3	0	1.5
d	0	3.5	0	2	2	3.5	2	1.5	0

(b)

FIG. 1 – (a) Problème à 4 sommets et 2 demandes (symbolisées par les flèches en pointillés) ; (b) Matrice des coûts (*Dist*) associée après duplication des sommets

2.3 Liens labellisés, tournées et tournées valides

On représente le déplacement du véhicule V d'un nœud x à un nœud y de X grâce à un lien labellisé : un lien labellisé est un triplet $r = (x, y, k)$, où x et y sont dans X et k est un label de l'ensemble $\{0\} \cup K$, x (resp. y) est noté $Start(r)$ (resp. $End(r)$) et k est appelé le label de r, noté $Label(r)$. La signification de k est alors que V est vide si $k = 0$ et contient la charge associée à la demande k sinon. Une tournée définie sur X est donc une suite Γ de liens labellisés.

Pour une telle tournée Γ et un label $k \in \{0\} \cup K$, on note Γ_k la suite des liens labellisés qui dérive naturellement de Γ en considérant dans Γ uniquement les liens labellisés r tels que $Label(r) = k$.

Le coût de Γ est donné par $Cout\_tournee(\Gamma) = \sum_{r \in \Gamma} Dist(Start(r), End(r))$.

Il est clair qu'une tournée ainsi définie ne représente pas forcément une solution du **SCPPA**. Il faut pour cela qu'elle représente le déplacement d'un véhicule qui traite toutes les demandes $k \in K$ de manière appropriée. Une telle tournée est dite **valide**, ce qui signifie que :
- pour chaque couple $(r, r' = Succ_\Gamma(r))$ de liens labellisés consécutifs dans Γ on a $End(r) = Start(r')$;
- $Start(First(\Gamma)) = End(Last(\Gamma)) = Depot$;
- chaque nœud $x \in X_O \cup X_D$ appartient à 2 liens labellisés r et $r' = Succ_\Gamma(r)$;
- le nœud $Depot$ appartient uniquement à $First(\Gamma)$ et $Last(\Gamma)$;
- pour chaque demande $k \in K$ la sous-tournée Γ_k associée est telle que :
  - $Start(First(\Gamma_k)) = o_k$;
  - $End(Last(\Gamma_k)) = d_k$;
  - pour chaque couple $(r, r' = Succ_{\Gamma_k}(r))$ de liens labellisés consécutifs dans Γ_k on a $End(r) = Start(r')$.

La figure 2 montre une tournée valide $\Gamma = \{$ ($Depot$, o_1, 0), (o_1, x, 1), (x, o_2, 0), (o_2, y, 2), (y, o_3, 0), (o_3, x, 3), (x, d_1, 1), (d_1, y, 0), (y, d_2, 2), (d_2, 0, x), (x, d_3, 3), (d_3, $Depot$, 0)$\}$. Les valeurs sur les arcs indiquent l'ordre de parcours.

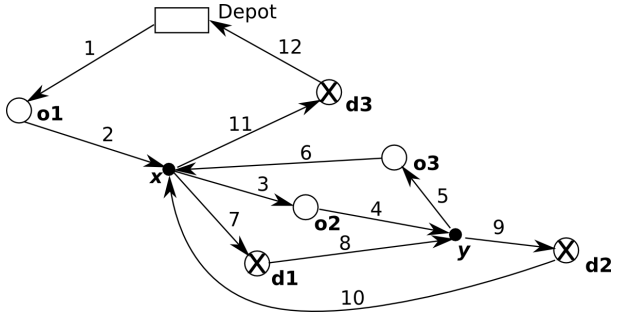


FIG. 2 – Exemple de tournée valide

Le **SCPPA** peut alors être posé de la manière suivante : étant donnés l'ensemble des nœuds X et la matrice des coûts $Dist$, calculer une tournée valide Γ de coût minimal.

3 Résultats structuraux

Le **SCPPA** est un problème difficile à traiter en se basant sur la représentation usuelle de la notion de tournée du fait des contraintes qui pèsent sur ces tournées et de l'hypothèse de préemption. Nous allons ici établir quelques résultats qui permettront de nous ramener à un problème moins contraint dans lequel l'objet inconnu est un arbre.

3.1 Théorème de restriction

Soit Γ une tournée valide. Pour chaque lien labellisé $r = (x, o_k, 0)$ dans Γ, nous notons $\sigma_\Gamma(r)$ l'unique lien labellisé $(d_k, y, 0)$ également dans Γ. De même, si x est un nœud de X_R tel qu'il existe un lien labellisé $r = (y, x, k), k \geq 1$ dans Γ, alors on notera $\sigma_\Gamma(r)$ le premier triplet $r' = (x, z, k)$ qui existe dans Γ_k. Remarquons que $\sigma_\Gamma(r)$ n'est défini que pour certains liens r (ceux de la forme $(x, o_k, 0)$ et ceux de la forme $(y, x, k), k \geq 1$).

On dira que deux liens labellisés r et r', pour lesquels $\sigma_\Gamma(r)$ et $\sigma_\Gamma(r')$ sont définis, **se recouvrent** si $Rg_\Gamma(\sigma_\Gamma(r')) > Rg_\Gamma(\sigma_\Gamma(r)) > Rg_\Gamma(r') > Rg_\Gamma(r)$.

On peut alors établir le *théorème de restriction* dont la signification est que l'on peut restreindre la recherche de solutions optimales pour le **SCPPA** à un sous-domaine dont les éléments sont des tournées valides dotées de propriétés additionnelles qui nous permettent de les représenter de manière très simple.

Théorème 1 (Théorème de restriction) *Soit Γ une tournée optimale pour le **SCPPA**, telle que :*

- *(A) : $|\Gamma|$ est le plus petit possible ;*
- *(B) : Le nombre de liens labellisés r dans Γ tels que $Label(r) \neq 0$, est le plus petit possible, (A) étant supposé satisfait.*

Alors les 4 assertions suivantes doivent être satisfaites :

- *(S1) : Γ ne doit pas contenir 2 occurrences du même lien labellisé $r = (x, y, k)$, avec $k \neq 0$;*
- *(S2) : Γ ne doit pas contenir 2 liens labellisés consécutifs r et r' tels que $Label(r) = Label(r')$;*
- *(S3) : Γ ne doit pas contenir 2 liens labellisés r et r' qui se recouvrent ;*

– *(S4) :* Γ *ne doit pas contenir 2 liens labellisés* r *et* r' *tels que* $End(r) = End(r')$ *et qui soient tous deux de label non nul.*

Nous ne fournissons pas la preuve de ce résultat ici (voir [29] pour la démonstration).

3.2 *Arbre-Reformulation* du SCPPA

Le théorème précédent nous conduit à réduire nos recherches lorsqu'on traite le **SCPPA** à des tournées valides qui satisfont les propriétés (S1) à (S4). Nous nommerons de telles tournées des tournées **fortement valides**. La figure 3 (a) montre la tournée de la figure 2 transformée en tournée fortement valide sans aucune perte au niveau du coût.

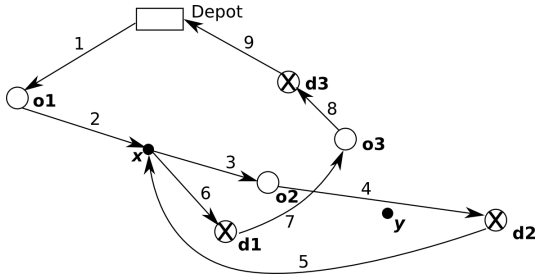


FIG. 3 – Tournée de la figure 2 transformée en tournée fortement valide

Ainsi résoudre le **SCPPA** revient à trouver une tournée fortement valide et de coût minimum.

Nous allons voir maintenant comment une tournée fortement valide peut être représentée comme un arbre, ce qui nous fournira les bases pour l'algorithme résolvant le **SCPPA** que nous introduirons dans la partie suivante.

On appelle **arbre biparti ordonné** un arbre T tel que :
– Ses nœuds peuvent être ordonnés en deux classes A et B de telle manière que la classe A a ses fils dans la classe B et vice versa ;
– Pour chaque nœud x dans T qui n'est pas une feuille (nœud terminal) l'ensemble des fils associés à x est ordonné linéairement et décrit sous forme de liste.

On dira qu'un arbre T biparti ordonné est **consistant** avec le **SCPPA** défini par l'ensemble K des demandes et par l'ensemble X des nœuds ((X, K)-consistant) si :
– Un nœud dans T peut être identifié soit par une demande $k \in K$ (nous pourrons ainsi parler d'un nœud *demande*), soit par un nœud dans $\{Depot\} \cup X_R$ (nous parlerons alors d'un nœud *relais*). Notons que tous les nœuds *demande* existants doivent apparaître dans T alors qu'il peut n'y avoir que quelques (voire aucun) nœuds parmi les nœuds *relais* dans T. Les nœuds *relais* de T forment l'ensemble des nœuds *relais actifs* de T, noté $Actif(T)$;
– La racine de T est le *Depot* et les feuilles sont des nœuds *demande* ;
– Pour chaque demande k, son ensemble de fils linéairement ordonné $Relais_T(k)$ (qui peut-être vide) est composé de nœuds *relais* actifs et son père $Pere_T(k)$ est dans $Actif(T)$;
– Pour chaque nœud *relais* x, son ensemble de fils linéairement ordonné $Demande_T(x)$ est composé de nœuds *demande* et son père $Pere_T(x)$ est dans K.

Pour un tel arbre biparti ordonné T, on peut définir un coût $Cout\_arbre(T)$ de la manière suivante :

– Pour chaque nœud *demande* $k \in K$, on pose :

$$\text{Si k est une feuille } Cout_{Demande,T}(k) = \quad Dist(o_k, d_k)\ ,$$
$$\text{sinon } Cout_{Demande,T}(k) = \quad Dist(o_k, First(Relais_T(k))) + Dist(Last(Relais_T(k)), d_k) + \sum_{\substack{x \in Relais_T(k),\\ x \neq Last(Relais_T(k))}} Dist(x, Succ_{Relais_T(k)}(x))\,;$$

– Pour chaque nœud $x \in \{Depot\} \cup X_R$, on pose :

$$Cout_{Relais,T}(x) = \quad Dist(x, o_{First(Demande_T(x))}) + Dist(d_{Last(Demande_T(x))}, x) + \sum_{\substack{k \in Demande_T(x),\\ k \neq Last(Demande_T(x))}} Dist(d_k, o_{Succ(Demande_T(x),k)})\,;$$

– Finalement le coût de T s'écrit : $Cout\_arbre(T) = \sum_{k \in K} Cout_{Demande,T}(k) + \sum_{x \in Actif(T)} Cout_{Relais,T}(x)$.

En exemple, on donne les coûts associés à l'arbre représenté figure 4 (b) :
$Cost_{Demande,T}(1) = Dist(o_1, x) + Dist(x, d_1)\,;$
$Cost_{Demande,T}(2) = Dist(o_2, d_2)\,;$
$Cost_{Demande,T}(3) = Dist(o_3, d_3).$
Le coût des relais s'écrit :
$Cost_{Relais,T}(x) = Dist(x, o_2) + Dist(d_2, x)\,;$
$Cost_{Relais,T}(Depot) = Dist(Depot, o_1) + Dist(d_3, Depot) + Dist(d_1, o_3).$

Les résultats suivants vont permettre de transformer le **SCPPA** en un problème de recherche d'arbre spécifique, plus facile.

Théorème 2 *Il y a une bijection nommée Arbre entre les tournées fortement valides et les arbres bipartis ordonnés (X, K)-consistants. De plus, pour toute tournée fortement valide Γ associée à un arbre T on a $Cout\_arbre(T) = Cout\_tournee(\Gamma)$.* Voir [29] pour la démonstration de ce théorème.

Une tournée fortement valide et l'arbre biparti associé sont représentés figure 4.

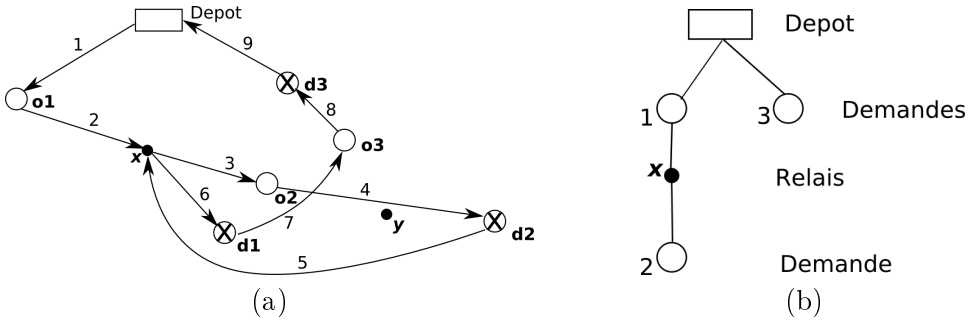


FIG. 4 – (a) Tournée fortement valide ; (b) Arbre associé

Corollaire 1 *Résoudre un problème **SCPPA** signifie trouver un arbre biparti ordonné T consistant avec X, K et tel que $Cout\_arbre(T)$ est le plus petit possible.*

L'intérêt de ce dernier résultat est clairement de nous fournir une formulation du **SCPPA** grâce à des arbres bipartis, cette formulation étant moins contrainte que le problème original. On notera que les fils d'une demande correspondent aux sommets *relais* sur laquelle celle-ci est rechargée. Les fils d'un sommet *relais* actif x correspondent aux demandes transportées sur le circuit partant de x.

3.3 Formulation du SCPPA avec un programme linéaire en nombres entiers

Construisons tout d'abord un graphe auxiliaire $G = (X^*, E)$:
– $X^* = X \cup X_R^* \cup \{Depot^*\}$, où X_R^* est une copie de X_R et $Depot^*$ est une copie de $Depot$;
Pour chaque nœud x dans X_R, on note x^* sa copie dans X_R^*. De même, pour chaque origine $x = o_k$ dans X_O, on note x^* le nœud d_k associé dans X_D.
– $E = \{(Depot, x), x \in X_O\} \cup \{(x, Depot^*), x \in X_D\} \cup \{(o_k, d_k), k \in K\} \cup \{(d_k, o_k), k \neq k' \in K\}$ $\cup \{(x, y), (y, x), x \in X_O, y \in X_R\} \cup \{(x, y)(y, x), x \in X_R^*, y \in X_D\} \cup \{(x, y), x \in X_R^*, y \in X_R\}$.
Chaque arc $e = (x, y) \in E$ possède une longueur $Dist^*(e) = Dist(x, y)$.

Nous rappelons qu'un chemin γ d'un tel graphe G est une suite de nœuds telle que, pour chaque nœud x dans γ, le couple $(x, Succ_\gamma(x))$ définit un arc de E. Une tournée fortement valide Γ peut être transformée en un chemin Γ^* de G de telle sorte que :
– (S5) : Γ^* commence en $Depot$ et finit en $Depot^*$ et Γ^* est un chemin élémentaire c.-à-d. qu'il visite chaque nœud au plus une fois ;
– (S6) : pour chaque demande $k \in K$, Γ^* visite o_k et d_k dans cet ordre, et pour chaque relais $x \in X_R$, Γ^* visite x si et seulement s'il visite x^*, et dans ce cas, il le fait dans cet ordre ;
– (S7) : pour chaque couple (x, y), $x \neq y$, dans $X_R \cup X_O$ l'implication suivante doit être vraie :
$Rg_{\Gamma^*}(x) < Rg_{\Gamma^*}(y)$ ET $Rg_{\Gamma^*}(y) < Rg_{\Gamma^*}(x^*) \Rightarrow Rg_{\Gamma^*}(y^*) < Rg_{\Gamma^*}(x^*)$
On appelle cette condition la *condition de non recouvrement* ;
– (S8) : $Cout\_Tournee(\Gamma) = \sum_{x \in \Gamma^*, x \neq Depot^*} Dist(x, Succ_{\Gamma^*}(x))$.

Il vient alors à partir du théorème 1, qu'une tournée fortement valide peut être représentée d'une manière qui nous permette d'établir un programme linéaire en nombres entiers. Ce modèle implique un vecteur flot $z = (z_e, e \in E)$ à valeur dans $\{0; 1\}$, un vecteur rang $R = (R_x, x \in X)$ à valeurs entières, et un vecteur de décision t indexé sur les couples $(x, y), x \neq y, (x, y) \in X_R \cup X_O$ à valeur dans $\{0; 1\}$. Ces vecteurs ont la sémantique suivante :
– $\forall e \in E, z_e = 1$ ssi l'arc e est dans γ ;
– $\forall x \in \gamma, R_x$ est égal au rang de x dans γ ;
– $\forall (x, y), x \neq y, (x, y) \in X_R \cup X_O$:

$t_{x,y} = 1$ ssi $Rg_\gamma(y) < Rg_\gamma(x^*)$;

$t_{x,y} = 0$ ssi $Rg_\gamma(x^*) < Rg_\gamma(y)$;

On peut alors traduire les propriétés $(S5) \ldots (S8)$ à l'aide de ces vecteurs et on obtient le programme linéaire suivant (pour chaque contrainte la propriété qu'elle traduit est notée entre parenthèses) :

Vecteurs inconnus :
– $z = (z_e, e \in E)$, à valeur dans $\{0, 1\}$;
– $R = (R_x, x \in X)$, entier positif ;
– $t = (t_{x,y}, x \neq y, (x, y) \in X_R \cup X_O)$, à valeur dans $\{0, 1\}$;

Critère de performance :
– Minimiser $\sum_{e \in E} DIST^*(e).z_e$ *(S8)* ;

Contraintes :
– z est un vecteur flot, qui satisfait les lois de Kirchhoff à chaque nœud sauf en $Depot$ et $Depot^*$;
– Le flot entrant, induit par z, en $Depot$ (respt. $Depot^*$) est égal à 0 (respt. 1), tandis que le flot sortant associé est égal à 1 (respt. 0) *(S5) et (S6)* ;
– Dans chaque nœud de $X_D \cup X_O$, le flot entrant induit par z est égal à 1 *(S5) et (S6)* ;
– Dans chaque nœud de $X_R \cup X_R^*$, le flot entrant induit par z est au moins égal à 1, et le flot

entrant en x est égal au flot entrant en x^* *(S5) et (S6)* ;
- Pour chaque demande $k \in K$, on a $R_{o_k} \leq R_{d_k} - 1$ *(S6)* ;
- Pour chaque relais $x \in X_R$, on a $R_x \leq R_{x^*} - 1$ *(S6)* ;
- Pour chaque arc $e = (x, y) \in E$:

  $z_e + (R_x + 1 - R_y)/|X^*| \leq 1$ (traduit l'implication $z_e = 1 \Rightarrow R_x + 1 - R_y \leq 0$) ;
- Pour chaque couple $(x, y), x \neq y, x, y \in X_R \cup X_O$ *(S7)* :

  $t_{x,y} + (R_y + 1 - R_{x^*})/|X^*| \leq 1$ (traduit l'implication $t_{x,y} = 1 \Rightarrow R_y + 1 - R_{x^*} \leq 0$ (I_1)).

  $t_{x,y} + (R_y - 1 - R_{x^*})/|X^*| \geq 0$ (traduit l'implication $t_{x,y} = 0 \Rightarrow R_y - 1 - R_{x^*} \geq 0$ (I_2)) ;

  $t_{x,y} + (R_{y^*} + 1 - R_{x^*})/|X^*| \leq 1$ (traduit l'implication $t_{x,y} = 1 \Rightarrow R_{y^*} + 1 - R_{x^*} \leq 0$ (I_3)).

Notons que le vecteur t sert à traduire la condition de non recouvrement. Les deux premières inégalités (I_1) et (I_2) donnent la sémantique de t : $t_{x,y} = 1$ signifie $R_y < R_{x^*}$ (I_1) et $t_{x,y} = 0$ signifie $R_{x^*} < R_y$ (I_2). La troisième inégalité (I_3) interdit la configuration suivante : $R_x < R_y < R_{x^*} < R_{y^*}$ dans laquelle on a recouvrement. En effet si $R_y < R_{x^*}$, il vient de (I_3) que $R_{y^*} < R_{x^*}$. De même, en inversant les rôles de x et de y, on obtient $R_x < R_{y^*} \Rightarrow R_{x^*} < R_{y^*}$. Donc si on a $R_y < R_{x^*}$ (I_3) donne que les seuls rangs possibles pour y^* et x sont tels que : $R_y < R_{y^*} < R_x < R_{x^*}$ ou $R_x < R_y < R_{y^*} < R_{x^*}$.

Remarquons également que (I_1) et (I_3) sont redondantes. En effet si $R_{y^*} < R_{x^*}$ alors $R_y < R_{x^*}$ car par définition $R_y < R_{y^*}$.

4 Heuristiques à base d'arbres pour le SCPPA

Notation préliminaire : L'opérateur d'affectation sera noté $\leftarrow$. Ainsi $x \leftarrow y$ signifie que la variable x reçoit la valeur de y. Dans un souci de simplification on appelera simplement *relais* un nœud *relais* et *demande* un nœud *demande*.

Les algorithmes décrits et testés ici se déduisent facilement de la représentation à l'aide d'arbres des solutions optimales du **SCPPA** obtenues dans la partie 3.2. Ces algorithmes sont de simples algorithmes gloutons et algorithmes de descente basés sur deux types d'opérateurs : opérateurs d'insertion et opérateurs de transformation locale.

4.1 Solution initiale

Les opérateurs d'insertion opèrent sur un arbre biparti ordonné T, consistant avec l'ensemble des nœuds X et avec un sous-ensemble K' de l'ensemble des demandes K. T est alors un arbre partiel comprenant uniquement les demandes de K'. Ces opérateurs insèrent une demande $k \in K \setminus K'$ dans T. On utilisera deux opérateurs :

- `Insert-simple` : ses paramètres sont un relais actif x dans $\{Depot\} \cup X_R$, et une coupe (l_1, l_2) de la séquence $Demande_T(x) = l_1 \oplus l_2$. Il insère la demande k dans cette coupe : $Demande_T(x) \leftarrow l_1 \oplus \{k\} \oplus l_2$. Cet opérateur consiste à insérer la demande k dans la tournée. Celle-ci est alors transportée sans relais après la dernière demande de l_1 et avant la première demande l_2 ;
- `Insert-avec-relais` : ses paramètres sont une demande $k' \in K'$, une coupe $c = (l_1, l_2)$ de la suite $Relais_T(k')$, et un relais non actif x. Cet opérateur commence par insérer le relais $\{x\}$ dans la coupe c : $Relais_T(x) \leftarrow l_1 \oplus \{x\} \oplus l_2$. Par conséquent x devient actif. Puis il insère la demande k : $Demande_T(x) \leftarrow \{k\}$. Cet opérateur consiste à insérer la demande k dans la tournée. Pour

cela, le véhicule décharge la demande k' au sommet x, va ensuite à l'origine o_k de la demande k et la transporte à sa destination d_k. Il revient enfin au sommet x pour recharger la demande k' et continue son trajet.

Nous pouvons alors proposer un premier algorithme `SCPPA-Insertion` (voir l'algorithme 1) d'insertion glouton, basé sur une arbre-représentation du problème, pour résoudre le **SCPPA** : l'arbre T est construit en utilisant les opérateurs d'insertion. T est tout d'abord initialisé à la racine $\{Depot\}$. Les demandes sont insérées séquentiellement (dans un ordre aléatoire), en choisissant à chaque fois la meilleure insertion possible dans l'arbre partiel T au sens de $Cout\_arbre(T)$. Pour éviter des tests inutiles nous utilisons un système de filtrage afin de s'orienter plus rapidement vers les jeux de paramètres u intéressants. Cela peut être fait en utilisant, pour chaque couple de nœuds (x, y) dans X_R, des ensembles $V(x)$ et $V(y)$ de voisins de x et y, le milieu z de x et y et en imposant des conditions entre ces variables au moment où l'on calcule les différents composants de u.

Algorithme 1 : **SCPPA**-Insertion

Définir aléatoirement un ordre ρ sur les demandes de K ;
$T = \{Depot\}$;
Pour $k \in K$ suivant l'ordre ρ **faire**
  Choisir un opérateur d'insertion I et le jeu de paramètres u associé tels que l'insertion de k dans T via $I(u)$ induise une augmentation du coût $Cout\_arbre(T)$ la plus faible possible ;
  Appliquer $I(u)$ à T ;

Il est clair que l'algorithme 1 d'insertion s'intègre parfaitement dans un schéma de type Monte-Carlo décrit par l'algorithme 2.

Algorithme 2 : **SCPPA**-Insertion couplé à un schéma de type Monte-Carlo

Paramètre : Δ ;
Pour *i = 1 à* Δ **faire**
  Exécuter SCPPA-Insertion ;
Garder le meilleur résultat ;

4.2 Algorithme de descente

Les opérateurs de transformation locale opèrent sur un arbre biparti ordonné T, consistant avec l'ensemble des nœuds X et l'ensemble des demandes K et ils le modifient. On utilisera six opérateurs :

– `Remplace-relais` : ses paramètres sont un relais actif x et un relais non actif x'. Il remplace x par x' ;

– `Deplace-relais` : ses paramètres sont deux demandes k et k', un segment l de $Relais_T(k)$ et une coupe $c = (l_1, l_2)$ de $Relais_T(k')$. Il supprime l de $Relais_T(k)$ et l'insère dans la coupe c. On suppose pour utiliser cet opérateur que k ne **domine** pas k' dans T c'est-à-dire k ne doit pas pouvoir être obtenu à partir de k' grâce à une suite d'applications de l'opérateur $Pere$;

– `Deplace-relais-l` : ses paramètres sont une demande k, un segment l de $Relais_T(k)$ tel que $Relais_T(k) = l_3 \oplus l \oplus l_4$ et une coupe $c = (l_1, l_2)$ de $l_3 \oplus l_4$. Cet opérateur supprime l de $Relais_T(k)$ et le réinsère dans la coupe c : $Relais_T(k) \leftarrow l_1 \oplus l \oplus l_2$;

– `Deplace-demande` : ses paramètres sont deux relais actifs différents x et x', un segment l de $Demande_T(x)$ et une coupe $c = (l_1, l_2)$ de $Demande_T(x')$. Il supprime l de $Demande_T(x)$ et l'insère dans la coupe c. Si $Demande_T(x)$ devient vide, le relais x est alors supprimé de T et devient non actif. On suppose pour utiliser cet opérateur que x ne domine pas x' ;

– `Deplace-demande-l` : ses paramètres sont un relais x, un segment l de $Demande_T(x)$ tel que $Demande_T(x) = l_3 \oplus l \oplus l_4$ et une coupe $c = (l_1, l_2)$ de $l_3 \oplus l_4$. Cet opérateur supprime l de $Demande_T(x)$ et le réinsère dans la coupe c : $Demande_T(x) \leftarrow l_1 \oplus l \oplus l_2$;

– `Deplace-demande-relais` : ses paramètres sont un relais actif x, un relais non-actif y , une demande k, un segment l de $Demande_T(x)$ et une coupe $c = (l_1, l_2)$ de $Relais_T(k)$. l est supprimé de $Demande_T(x)$, inséré dans $Demande_T(y)$ (y devient donc actif), y étant inséré dans c. Si $Demande_T(x)$ devient vide (cas où $l = Demande_T(x)$) alors x devient non actif. On suppose pour utiliser cet opérateur que la demande k n'est dominée par aucune des demandes k' dans l.

Les opérateurs de transformation locale nous permettent de construire un algorithme de descente `SCPPA-Descente` décrit par l'algorithme 3 : on construit tout d'abord un arbre T avec `SCPPA-Insertion`. Ensuite on cherche à l'améliorer (c'est-à-dire réduire $Cout\_arbre(T)$) en appliquant des transformations locales successives à l'aide des opérateurs décrits précédemment (voir instruction ($I1$) dans l'algorithme 3). L'efficacité de l'algorithme dépend en grande partie du choix de l'opérateur I à appliquer et de son jeu de paramètres u. Le nombre de choix envisageables étant très grand, on évite de tester tous les paramètres possibles pour chaque opérateur : on utilise donc un mécanisme de filtrage similaire à celui proposé dans le cadre de `SCPPA-Insertion`. On introduit de plus une valeur seuil H tel que plus H est petit plus le domaine de recherche des paramètres est grand. H décroît alors au fur et à mesure des itérations permettant ainsi d'agrandir l'espace de recherche à chaque fois que la recherche dans l'espace restreint par H a échoué.

Algorithme 3 : **SCPPA**-Descente

Initialiser l'arbre T avec **SCPPA**-Insertion ;
Initialiser la valeur seuil filtrante H ;
stop $\leftarrow$ faux ;
tant que *stop* = *faux* **faire**
  Rechercher (en utilisant le filtre H) un opérateur I et un jeu de paramètres u tels que l'application de $I(u)$ sur T réduise $Cout\_arbre(T)$ ($I1$) ;
  si *la recherche a échoué* **alors** $H \leftarrow H/2$;
  si H *devient trop petit* **alors** stop $\leftarrow$ vrai ;

5 Résultats expérimentaux

Les algorithmes décrits dans la section précédente ont été implémentés en C++ (le compilateur utilisé est celui de Visual Studio 2008), testés sur PC Intel Xeon, 1.86GHz, 3.25 GO de Ram.

Nous nous sommes concentrés sur plusieurs points :
– la capacité de `SCPPA-Insertion` et `SCPPA-Descente` à trouver de manière rapide des solutions proches de l'optimal théorique ;
– les caractéristiques des solutions : nombre de relais dans la solution (c'est-à-dire impact de l'hypothèse de préemption).
Dans ce but nous avons réalisé plusieurs tests. Nous avons utilisé les instances de la librairie TSPLIB (voir http ://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) pour fournir les ensembles

de nœuds X et la matrice de distance $Dist$, les couples origine / destination étant choisis aléatoirement parmi les nœuds de X. Nous avons traité des instances comportant de 20 à 300 nœuds et de 10 à 100 couples origine / destination. Les instances aléatoires ont été résolues de manière exacte par un algorithme de coupes et branchements à l'aide d'un programme linéaire similaire à celui donné dans la section 3.3.

Nous avons tout d'abord testé l'algorithme `SCPPA-Insertion` couplé au schéma Monte-Carlo avec $\Delta = 100$. Les résultats sont présentés dans le tableau 1.

Nous utilisons les notations suivantes :
- *dem.* : nombre de demandes ;
- *som.* : nombre de sommets ;
- *inst.* : nom de l'instance ;
- *ref* : valeur de la solution optimale obtenue grâce au programme linéaire ;
- *min* : valeur de la meilleure solution obtenue parmi les Δ itérations de `SCPPA-Insertion` ;
- *écart* : écart en % entre *ref* et *min* ;
- *rel.* : nombre moyen de relais actifs dans les solutions produites par `SCPPA-Insertion` ;
- *cpu* : temps moyen en millisecondes d'une exécution de `SCPPA-Insertion`.

dem.	*som.*	*ref.*	*min.*	*écart (%)*	*rel.*	*cpu (ms)*
11	46	24 654	25 023	1.5	0.06	< 15
11	46	21 395	21 424	0.14	0.43	< 15
23	94	318 959	325 027	1.9	0.95	< 15
23	94	315 118	322 908	2.47	0.53	15
59	238	318 099	337 259	6.02	0.36	78

TAB. 1 – Tests de `SCPPA-Insertion` couplé au schéma Monte-Carlo avec $\Delta = 100$, réalisés sur 5 instances obtenues à partir des instances de la TSPLIB en choisissant aléatoirement les demandes (tous les nœuds, hormis le dépôt, sont donc soit l'origine d'une demande, soit la destination et les relais doivent être choisis parmi ceux-ci)

Le tableau 2 montre les résultats sur des instances *RELn* que nous avons construites de façon à ce que la solution optimale comporte des relais actifs, testant ainsi la capacité de l'algorithme à les trouver. Elles comportent les caractéristiques suivantes :
- l'instance comporte n nœuds, $n/2$ relais et $((n/2) - 1)/2$ demandes ;
- sa valeur optimale est $\sum_{k \in K} Dist(o_k, d_k)$, et la solution optimale est telle que tous les relais sont utilisés.

La figure 5 visualise sur un plan une tournée solution d'une instance de type *RELn* (avec $n = 26$), ainsi que son arbre associé. En chaque point $\{A \dots G\}$ du plan sont positionnés différents nœuds de l'ensemble X : les nœuds origines sont notés $o_i, i = 1 \dots 6$, les nœuds destinations sont notés $d_i, i = 1 \dots 6$ et le nœud relais représenté est l'unique relais actif noté r. Les valeurs sur les arcs de la tournée indique l'ordre de parcours.

Cette heuristique nous permet d'obtenir des solutions d'assez bonne qualité de manière très rapide. Cependant on remarque que l'heuristique est de moins en moins efficace à mesure que le nombre de relais actifs augmente.

Le second test consiste à exécuter `SCPPA-Descente` à partir d'une solution fournie par `SCPPA-Insertion`. On notera :
- *dem.* : nombre de demandes ;
- *som.* : nombre de sommets ;

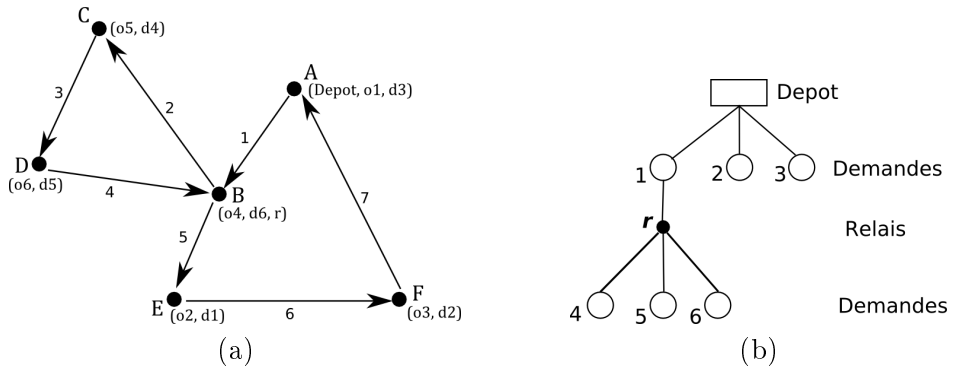


FIG. 5 – (a) Tournée solution d'une instance "*RELn*" ; (b) Arbre associé

inst.	*ref.*	*min.*	*écart(%)*	*rel.*	*cpu (ms)*
REL62	11 843	13 045	10.1	0.11	< 15
REL110	17 464	19 554	12.0	0.35	16
REL158	21 053	24 235	15.1	0.54	31
REL182	22 323	26 321	17.9	0.54	47
REL230	24 142	28 117	16.5	0.91	78
REL326	26 036	30 734	18.0	1.76	141
REL374	26 494	32 077	21.1	2.87	187
REL422	26 775	32 830	22.6	3.39	250
REL518	27 042	33 773	24.9	5.92	375
REL566	27 098	33 664	24.2	6.86	453

TAB. 2 – Tests de `SCPPA-Insertion` effectués sur 10 instances "RELn"

- *ref* : valeur de la solution optimale obtenue grâce au programme linéaire ;
- *min* : valeur de la solution obtenue après l'application `SCPPA-Descente` ;
- *écart 1* : écart en % entre *ref* et la valeur de la solution initiale produite par `SCPPA-Insertion` ;
- *écart 2* : écart en % entre *ref* et *min* ;
- *rel.* : nombre de relais actifs dans la solution produite par `SCPPA-Descente` ;
- *cpu* : temps moyen en secondes de l'exécution de `SCPPA-Descente`.

dem.	*som.*	*ref.*	*min.*	*écart 1 (%)*	*écart 2 (%)*	*rel.*	*cpu (s)*
11	46	24 654	24 654	10.3	0	1	0.1
11	46	21 395	21 914	5.0	2.4	1	0.05
23	94	318 959	322 879	5.7	1.2	2	2.0
23	94	315 118	316 362	5.1	0.4	5	0.7
59	238	318 099	31 913	9.5	0.25	4	103

TAB. 3 – Tests de `SCPPA-Descente` ; réalisés sur 5 instances obtenues à partir des instances de la TSPLIB en choisissant aléatoirement les demandes (tous les nœuds, hormis le dépôt, sont donc soit l'origine d'une demande, soit sa destination, et les relais doivent être choisis parmi ceux-ci)

On peut tout d'abord remarquer que l'écart entre préemption et non préemption est très différent suivant la manière dont ont été générées les instances. Bien que notre algorithme de descente `SCPPA-Descente` n'utilise aucun des mécanismes habituels de contrôle destinés à gérer les optima locaux (recherche tabou, recuit simulé ...), on peut constater que les opérateurs qui dérivent de notre représentation en arbre fournissent de très bons résultats dans des temps de calcul très courts.

inst.	*ref.*	*min.*	*écart 1 (%)*	*écart 2 (%)*	*rel.*	*cpu (s)*
REL62	11 843	11 953	16.7	0.9	4	0.04
REL110	17 464	17 583	27	0.7	8	0.34
REL158	21 053	21 053	29.9	0	12	1.5
REL182	22 323	22 605	20.8	1.2	14	2.9
REL230	24 142	24 225	28.4	0.34	18	8.4
REL326	26 036	26 279	19.3	0.93	26	25
REL374	26 494	26 626	26.6	0.49	30	39
REL422	26 775	26 851	33.1	0.28	34	78
REL518	27 042	27 067	37	0.09	42	232
REL566	27 098	27 211	33.1	0.4	46	328

TAB. 4 – Tests de `SCPPA-Descente` effectués sur 10 instances "RELn"

6 Conclusion

Nous avons traité ici un problème de Ramassage et Livraison avec des demandes préemptives et des contraintes de capacité et nous avons montré qu'il était possible de le transformer en un problème de construction d'arbre peu contraint, de telle sorte que nous puissons le résoudre grâce à de simples et efficaces algorithmes de construction gloutonne et de descente. Il serait intéressant d'étendre cette approche présentée ici dans un contexte spécifique, et d'en déduire des approches efficaces pour traiter les problèmes de préemption dans des problèmes de tournées ou d'ordonnancement plus généraux.

Références

[1] S. Anily, M. Gendreau, G. Laporte. The preemptive swapping problem on a tree. Technical report, Les Cahiers du GERAD, G-2005-69, 2006.

[2] S. Anily, R. Hassin. The swapping problem. Networks, 22(4) :419-433, 1992.

[3] N. Ascheuer, L. Escudero, M. Grötschel, M. Stoer. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). SIAM Journal on Optimization, 3 :25-42, 1993.

[4] N. Ascheuer, M. Jünger, G. Reinelt. A Branch & Cut algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints. Computational Optimization and Applications, 17 :61-84, 2000.

[5] M.J. Atallah, S.R. Kosaraju. Efficient Solutions to Some Transportation Problems with Applications to Minimizing Robot Arm Travel. SIAM Journal on Computing, 17 :849, 1988.

[6] E. Balas, M. Fischetti, W. Pulleyblank. The precedence constrained asymmetric traveling salesman problem. Mathematical Programming, 68 :241-265, 1995.

[7] G. Berbeglia, J.F. Cordeau, I. Gribkovskaia, G. Laporte. Static pickup and delivery problems : a classification scheme and survey. TOP : An Official Journal of the Spanish Society of Statistics and Operations Research, 15(1) :1-31, July 2007.

[8] C. Bordenave, M. Gendreau, G. Laporte. A branch-and-cut algorithm for the preemptive swapping problem. Technical Report CIRRELT-2008-23, 2008.

[9] C. Bordenave, M. Gendreau, G. Laporte. Heuristics for the mixed swapping problem. Technical Report CIRRELT-2008-24, 2008.

[10] S. Chen, S.Smith. Commonality and genetic algorithms. Technical Report CMU-RI-TR-96-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1996.

[11] J.F. Cordeau, M. Iori, G. Laporte, J.J. Salazar-González. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Networks, 2008, in press. Voir http ://neumann.hec.ca/chairedistributique/common/tsppdl-bc.pdf.

[12] J.F. Cordeau, G.Laporte. The dial-a-ride problem : models and algorithms. Annals of Operations Research, 153(1) :29-46, 2007.

[13] C.E. Cortés, M. Matamala, C.Contardo. The Pickup and Delivery Problem with Transfers : Formulation and Solution Approaches. In VII French - Latin American Congress on Applied Mathematics. Springer, 2005.

[14] I. Dumitrescu. Polyhedral results for the pickup and delivery travelling salesman problem. Technical Report, CRT-2005-27, 2005.

[15] M.T. Fiala Timlin. Precedence constrained routing and helicopter scheduling. M. Sc. Thesis, Department of Combinatorics and Optimization University of Waterloo, 1989.

[16] M.T. Fiala Timlin, W.R. Pulleyblank. Precedence constrained routing and helicopter scheduling : heuristic design. Interfaces, 22 :100-111, 1992.

[17] G.N. Frederickson and D.J. Guan. Preemptive Ensemble Motion Planning on a Tree. SIAM Journal on Computing, 21 :1130, 1992.

[18] G.N. Frederickson and D.J. Guan. Nonpreemptive ensemble motion planning on a tree. Journal of Algorithms, 15(1) :29-60, 1993.

[19] G.N. Frederickson, M.S. Hecht, C.E. Kim. Approximation Algorithms for Some Routing Problems. SIAM Journal on Computing, 7 :178, 1978.

[20] L.M. Gambardella, M. Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing, 12(3) :237-255, 2000.

[21] L. Gouveia and P. Pesneau. On extended formulations for the precedence constrained asymmetric traveling salesman problem. Networks, 48(2) :77-89, 2006.

[22] H. Hernández-Pérez, J. Salazar-González. The multicommodity one-to-one pickup-and-delivery traveling salesman problem. European Journal of Operational Research, 196 :987-995 2009

[23] S. Mitrovic-Minic, G. Laporte. The pickup and delivery problem with time windows and transshipment. INFOR, 44 :217-227, 2006.

[24] R. Montemanni, D.H. Smith, L.M. Gambardella. A heuristic manipulation technique for the sequential ordering problem. Computers & Operations Research, 35(12) :3931-3944, 2008.

[25] P. Oertel. Routing with Reloads. Doktorarbeit, Universität zu Köln, 2000.

[26] S.N. Parragh, K.F. Doerner, R.F. Hartl. A survey on pickup and delivery problems : Part II : Transportation between pickups and delivery locations. Journal für Betriebswirtschaft, 58(1) :21-51, 2008.

[27] J. Renaud, F. Boctor, G. Laporte. Perturbation heuristics for the pickup and delivery traveling salesman problem. Computers & Operations Research, 29(9) :1129-1141, 2002.

[28] K.M. Ruland, E.Y. Rodin. The pickup and delivery problem : Faces and branch-and-cut algorithm. Computers and Mathematics with Applications, 33 :1-13, 1997.

[29] H. Kerivin, M. Lacroix, A. Quilliot, H. Toussaint. Heuristic for the preemptive asymmetric stacker crane problem. Rapport de recherche RR09-01, Laboratoire LIMOS, Université Blaise Pascal, Clermont-Ferrand, 2009.