8e Conférence Internationale de MOdélisation et SIMulation - MOSIM'10 - 10 au 12 mai 2010 - Hammamet - Tunisie
« Évaluation et optimisation des systèmes innovants de production de biens et de services »

Recherche locale pour un problème d'optimisation de tournées de véhicules avec gestion des stocks

T. Benoist, F. Gardi, A. Jeanjean

Bouygues e-lab
Paris, France
{tbenoist,fgardi,ajeanjean}@bouygues.com

B. Estellon

Laboratoire d'Informatique Fondamentale
Faculté des Sciences de Luminy, Marseille, France
bertrand.estellon@lif.univ-mrs.fr

RÉSUMÉ : *Nous abordons ici un problème d'optimisation de tournées de véhicules avec gestion des stocks à long terme. La particularité de ce problème est que le vendeur contrôle l'inventaire de ses clients, en décidant quand et de combien chaque stockage doit être approvisionné par les tournées de camions. L'objectif du vendeur est de minimiser ses coûts logistiques sur le long terme. Nous présentons tout d'abord la modélisation de ce problème tel que rencontré en pratique dans une grande entreprise française. Puis, nous introduisons une approche de résolution sur le court terme, effective pour optimiser l'objectif à long terme, basée sur une heuristique de recherche locale basée sur une fonction-objective de substitution utilisant des bornes inférieures. Une étude a permis de montrer que notre approche est à la fois efficace et robuste induisant des réductions de coûts de l'ordre de 20 % par rapport à des solutions construites par des planificateurs experts. Le logiciel incorporant ce moteur est désormais en exploitation.*

MOTS-CLÉS : *logistique ; tournées de véhicules ; gestion des stocks ; recherche locale.*

1 Tournées avec gestion des stocks

Le problème proposé dans cet article est un problème réel d'optimisation de tournées de véhicules avec gestion des stocks, communément appelé Inventory Routing Problem (IRP) en anglais. Par souci de concision, le problème ne sera pas complètement et formellement décrit ici, mais les principales caractéristiques y sont présentées.

Sur une zone géographique, on retrouve trois types de sites : *dépôts*, *clients* et *usines*. Les distances entre chaque point ne sont pas euclidiennes mais fournies en entrée du problème. Les *dépôts* auxquels sont associés des *ressources* (chauffeurs, remorques, tracteurs) est utilisé comme point de départ et d'arrivée de toute tournée. Concernant les clients, deux types d'approvisionnement sont gérés par le fournisseur. Le premier mode, appelé "à la précision", correspond aux clients pour lesquels on connaît des prévisions précises de consommation à court terme. Le fournisseur doit donc veiller à ce que chaque client soit re-approvisionné par camion de façon à ne jamais tomber sous son seuil de sécurité. Le deuxième mode qu'on appelle "à la commande" concerne les clients qui passent des commandes au fournisseur, en précisant qu'ils souhaitent être livrés d'une certaine quantité de produit pendant une fenêtre de temps. Des clients peuvent demander les deux types de politique d'approvisionnement. Ce type d'approvisionnement mixte consiste à ce que leur stock soit géré par le fournisseur en utilisant la surveillance et les prévisions, mais ils se gardent la possibilité de passer une commande (pour faire face à des augmentations imprévues de leur consommation par exemple). Les contraintes relatives au maintien du niveau de leur stock au-dessus du seuil de sécurité (pas d'assèchement) et à la satisfaction des commandes (pas de commande non traitée) sont définies comme souples, puisque l'existence d'une solution admissible n'est pas garantie dans des conditions réelles. Un *client* est représenté par sa capacité de stockage (la quantité maximale livrable), son seuil de sécurité (la quantité minimale de produit qui doit être maintenue dans le stock), sa quantité initiale de produit dans le reservoir au début de la période, ses prévisions de consommation pour chaque pas de temps, son ensemble de fenêtres de temps d'ouverture, les ensembles de chauffeurs, tracteurs et remorques autorisés à entrer sur le site, son temps fixe d'opération pour réaliser une livraison, sa liste de commandes passées, ses coûts associés à chaque commande non traitée et ceux associés à chaque pas de temps passé sous le seuil de sécurité. Un client peut avoir un attribut supplémentaire : il peut ou non accepter des livraisons non sollicitées (afin de savoir si ce client a ce type d'approvisionnement mixte ou non). Une *commande* est caractérisée par la quantité demandée par le client et la fenêtre de temps pendant laquelle la livraison doit avoir lieu. Une *usine* est définie de manière similaire à un client, sans les attributs relatifs aux commandes et

au niveau de sécurité. A noter que les consommations d'un client (resp. d'une usine de production) sont représentées par des valeurs positives (resp. négatives).

Les livraisons à ces clients sont assurées par des véhicules composés de trois types de ressources hétérogènes : des chauffeurs, des tracteurs et des remorques. Un *chauffeur* peut représenter un conducteur ou une paire de conducteurs (équipe en place dans les zones étendues pour des voyages très longs). Un chauffeur est donc défini par sa base de rattachement, ses disponibilités sur la période de planification modélisées par un ensemble de fenêtres de temps disjointes, l'ensemble des tracteurs qu'il peut conduire, son amplitude maximale de travail sur chaque tournée, sa durée maximale de conduite après laquelle une pause est requise, sa durée maximale de travail après laquelle une pause est aussi requise, sa durée minimale de pause, son coût horaire, ses coûts pour les opérations de chargement et livraison et enfin son coût fixe par pause. Un *tracteur* est quand à lui défini par sa base, ses disponibilités sur la période modélisées par un ensemble de fenêtres de temps disjointes, son ensemble de remorques compatibles, sa vitesse (un entier en 0 et 9 utilisé comme indice de la matrice de temps), son coût par unité de distance parcourue. Enfin, une *remorque* est elle aussi définie par sa base, son ensemble de fenêtres de temps de disponibilité ainsi que par sa capacité (la quantité maximale de produit qu'elle peut transporter) et sa quantité de produit présente au début de la période.

Dans notre modélisation, toutes les dates sont gérées à une granularité arbitraire (que nous avons fixée à la minute) : les dates de livraisons/chargements, les périodes de disponibilité des ressources, les heures d'ouverture des sites, les pauses. Cependant, les consommations de chaque client et les productions de chaque source sont connues par heure. Cette discrétisation du temps permet d'établir un inventaire à chaque pas de temps. Toutes les fenêtres de temps sont gérées de telle manière que la date de début est incluse dans l'intervalle et celle de fin est exclue. Deux matrices de données sont fournies : la première représente la distance entre chaque site tandis que la seconde le temps nécessaire pour parcourir la distance entre deux points en fonction d'un paramètre dont la valeur varie de 0 à 9 représentant la vitesse du tracteur. Ces deux matrices ne sont pas nécessairement symétriques mais elles respectent l'inégalité triangulaire. Au début et à la fin de chaque tournée ainsi qu'avant et après chaque pause, des temps de pré et post traitements viennent s'ajouter à la durée totale de la tournée.

Trouver une solution à ce problème va consister à définir un ensemble de tournées valides. Une *tournée* est une liste d'opérations effectuées par un triplet de ressources (chauffeur, tracteur et remorque). Elle commence et se termine à la base. Une *opération* est définie par sa tournée d'appartenance, son site où elle a lieu, son type (chargement/livraison), la quantité livrée ou chargée (positive pour les livraison, négative pour les chargements), la commande satisfaite par cette opération (si il y en a une) et ses dates de début et de fin (fenêtre de temps qui doit être incluse dans une période d'ouverture du site). Les sites visités (sources ou clients) durant la tournée doivent être accessibles à chacune des ressources composant le véhicule. L'intervalle de temps pendant lequel est utilisée une ressource doit être contenu dans une des fenêtres de disponibilité de celle-ci, et ce pour les trois ressources. Les tournées d'une même ressource ne peuvent pas se chevaucher dans le temps (les intervalles de temps des tournées sont deux à deux disjoints) et ne peuvent pas durer plus d'un certain nombre d'heures, dépendant du chauffeur. De plus, entre chaque tournée d'un même chauffeur, une pause de durée minimum fixée, est nécessaire. Les temps cumulés de travail et de conduite ne peuvent dépasser respectivement les temps maximal de travail et de conduite autorisés pour ce chauffeur. Le temps minimum pour se déplacer d'une opération à la suivante est donné par la matrice des temps de parcours. Il est parfois nécessaire d'attendre la prochaine fenêtre d'ouverture d'un site. Si ce temps d'attente peut être supprimé en l'intégrant à une pause ayant lieu au court de ce même déplacement, on décalera le départ du chauffeur après sa pause. Ceci permet de réduire le coût en comptabilisant ce temps comme du temps de pause et non plus comme du temps de travail.

En plus de ces contraintes de planification des tournées, le problème nécessite de satisfaire les contraintes d'inventaires. Les niveaux d'inventaires pour chaque client, usine et remorque sont calculables à partir des quantités livrées ou chargées. Bien évidement, ces inventaires doivent rester à tout pas de temps positifs et inférieurs à la capacité du stockage. A chaque pas de temps, la quantité présente dans le réservoir d'un client est constitué de la quantité du pas de temps précédent (ou de la quantité initiale si on est au premier pas de temps), augmentée de la quantité livrée et diminuée de la quantité de produit consommée au cours de ce pas de temps. Inversement pour les usines pour qui on diminue l'inventaire de la quantité chargée et on l'augmente de la quantité produite. Quand aux remorques, leur inventaire doit être maintenu en fonction des opérations de livraison, de chargement et de la quantité initiale présente au début de la période. Il est important de maintenir la cohérence de l'inventaire de la remorque en conservant les stocks restants d'une tournée à l'autre.

Dans notre cas, les prévisions fiables (aussi bien pour les usines que pour les clients) sont connues pour un horizon temporel de 15 jours. Ainsi, les tournées sont planifiées de façon déterministe jour après jour avec

un horizon glissant de 15 jours. Cela signifie que pour chaque jour, on construit un planning de distribution pour les 15 jours suivant, mais on conserve uniquement les tournées débutant le jour courant en les fixant.

L'objectif de ce problème, défini sur le long terme (une centaine de jours), est scindé en trois objectifs. Tout d'abord, on cherche à éviter les commandes non-satisfaites. Ensuite, le second objectif est de minimiser les assèchements. Enfin, le dernier objectif consiste à minimiser le ratio logistique. Les coûts de production ne sont pas pris en compte ici, mais uniquement les coûts de distribution. Le premier terme de la fonction objectif s'intéresse aux commandes non-satisfaites. Une commande est satisfaite si une opération est affectée avec une quantité livrée supérieure ou égale à la quantité attendue et une date d'arrivée inclue dans la fenêtre de temps escomptée. Pour chaque client c, on note *NC(c)* le nombre de commandes non-satisfaites et *CC(c)* le coût associé à chacune des ces commandes non livrées. On en déduit que le coût total des commandes non-satisfaites CC est donné par :

$$CC = \sum_{c \in Clients} CC(c) \times NC(c) \quad (1)$$

Le second coût optimisé ici concerne les assèchements. Un assèchement apparaît quand le niveau d'inventaire d'un client (ne fonctionnant pas "à la commande") se retrouve sous le seuil de sécurité pour un pas de temps donné. Pour chaque client c, on note *NA(c)* le nombre de pas de temps passés en assèchement et *CA(c)* le coût associé. On en déduit donc que le coût total des assèchement CA s'écrit :

$$CA = \sum_{p \in Clients} CA(c) \times NA(c) \quad (2)$$

Pour éviter les effets de bords de fin de planning, les commandes non-satisfaites et les assèchements sont comptabilisés sur l'horizon de temps raccourci T', qui correspond au l'horizon T diminué du maximum des amplitudes maximales (afin d'être sûr que les exigences qui se posent à la fin de l'horizon puissent toujours être satisfaites).

Le troisième et dernier terme est le ratio logistique $RL = CT/QT$ avec CT le coût total des tournées et QT la quantité totale livrée sur l'horizon considéré (sauf si $QT = 0$, alors $RL = 0$). Ainsi, QT vaut la somme des quantités livrées pour toutes les tournées. Le coût total d'une tournée CT se décompose en des coûts kilométriques (dépendant de la distance parcourue réalisée par le tracteur au cours de la tournée), des coûts temporels (dépendant du temps de travail dependant du chauffeur) et des coûts forfaitaires pour chaque livraison, chargement et pauses.

Il est important de voir que les trois termes CC, CA et RL de la fonction objectif sont optimisés dans un ordre lexicographique : $CC \succ CA \succ RL$ mais les solutions avec $CC = 0$ et $CA = 0$ peuvent être trouvées facilement en pratique.

Des instances de grande taille doivent être traitées. Une zone géographique peut contenir jusqu'à 1500 clients, 50 sources, 50 bases, 100 chauffeurs, 100 tracteurs et 100 remorques. Toutes les dates et les durées sont exprimées en minutes. En tout, le planning à horizon court terme compte 21600 minutes (Noter qu'on est capable ici de gérer un temps continu). Les inventaires des usines et des clients sont calculés avec des pas de temps d'une heure (car les prévisions sont calculées avec cette précision). Le temps d'exécution pour calculer un planning court terme est limité à 5 minutes sur des ordinateurs standards.

2 État de l'art et contributions

Bell et al, 1983, introduisent le problème d'IRP rencontré chez un industriel. Cette publication a donné naissance à de nombreux travaux dans ce domaine et en particulier une longue série de papiers publiés par Campbell et al. (1998, 2002), Campbell and Savelsbergh (2004a, 2004b, 2004c), Savelsbergh and Song (2007, 2008). Pour de nombreuses références sur le sujet, nous invitons le lecteur à consulter les papiers récents de Savelsbergh and Song (2007, 2008) qui offrent un résumé détaillé des travaux réalisés sur l'IRP ces 25 dernières années.

2.1 Contribution à la modélisation

Le problème décrit ici s'approche de près du problème traité quotidiennement par les planificateurs. De nombreuses contraintes inclues dans notre modélisation ne sont pas présentes dans les problèmes décrits dans la littérature. Tout d'abord, notre modèle prend en compte aussi bien les clients à la prévision que ceux à la commande. Ensuite, nous prenons en compte des consommations/productions non-linéaires (ces prévisions sont supposées connues et déterministes sur l'horizon). A notre connaissance, les seuls travaux publiés s'appuyant sur des données réelles sont ceux de Campbell et al. (2002), Campbell et Savelsbergh (2004), Savelsbergh et Song (2007, 2008). Par ailleurs, et contrairement à certains de ces travaux, nous considérons la durée de livraison fixe, c'est-à-dire ne dépendant pas du volume de livraison. Mais notre algorithme pourrait tout à fait être adapté pour le faire. Nous avons fixé ces paramètres car nous jugeons cette approximation suffisante et nous verrons dans la section suivante qu'on simplifie ainsi l'algorithmique sous-jacente. Une de nos contributions est ici l'introduction d'une fonction objectif de substitution que nous utilisons pour résoudre le problème de planification à court terme tout en ga-

rantissant une optimisation à long terme. La planification à court terme est construite pour 15 jours dans les détails mais on conserve uniquement les tournées débutant le premier jour avant de faire glisser l'horizon planning. Afin de maximiser le ratio logistique "coût par kilo de produit livré" sur le long terme, la fonction objectif prend en compte la distance parcourue, le temps de transport, le nombre de chargements et de livraisons, le nombre et la durée de pauses et la particularité est de considérer aussi un facteur objectif complémentaire basé sur des bornes inférieures obtenues au préalable pour chaque client.

2.2 Contribution à la résolution

Avant de présenter notre approche de résolution, nous allons détailler les résultats obtenus par Campbell et al. (2002) et Campbell et Savelsbergh (2004a). Les méthodes de résolution de ces deux papiers sont assez similaires : dans une première phase utilisant des techniques de programmation en nombre entiers, il est décidé quels clients sont livrés et quel est leur premier volume cible de livraison. Puis une deuxième étape résolue à l'aide d'une heuristique d'insertion permet de prendre en compte les contraintes de capacité des véhicules, des fenêtres d'ouverture, des restrictions de conduite, *etc.* Le premier problème (Campbell et al. 2002) décrit des plannings précis sur 5 jours et une vision agrégée sur 4 semaines. Les instances comportent 50, puis 87 clients avec 4 véhicules. La solution est comparée à celle obtenue à l'aide d'un algorithme glouton. Leur approche permet un gain de 8.11 % de volume par mile et une meilleure utilisation des ressources. Dans le deuxième papier (Campbell et Savelsbergh, 2004a) travaillent sur une période de 10 jours (3 jours détaillés et 7 jours agrégés), glissante sur 1 mois. De même, les résultats sont comparés à une approche gloutonne avec une limite de 10 minutes par itération, pour une instance comptant 100 puis 50 clients. Les gains mesurés sont de 2.70 % de volume par mile avec une meilleure utilisation des volumes des camions et des tournées plus courtes.

Nous présentons ici un algorithme de recherche locale pur et direct pour résoudre le problème de tournées de véhicules avec gestion des stocks. Par "pur", nous signifions qu'il n'y a aucune métaheuristique ni hybridation et "direct" signifie que l'approche du problème est globale, sans aucune décomposition. Une approche par recherche locale est décrite par Lau et al. (2002) pour résoudre ce même problème avec des fenêtres de temps, mais leur approche se base sur une décomposition. Dans ce papier, nous décrivons une heuristique originale de recherche locale pour résoudre le problème de planification à court terme. Le planning à court terme est optimisé directement sur les 15 jours de planning, sans aucune décomposition. Cet algorithme suit la méthodologie d'ingénierie algorithmique introduite par Estellon et al. (2009). Des routines, identifiées comme critiques pour les performances globales de l'algorithme, s'appuient sur des structures de données spécialement conçues pour exploiter les invariants des transformations. En moyenne, notre algorithme visite plus de 10 millions de solutions dans l'espace de recherche pendant les 5 minutes de temps de calcul, avec un taux de diversification de presque 5 % (c'est-à-dire le nombre de transformations validées sur le nombre d'évaluées), qui permet d'atteindre rapidement des optima locaux de grande qualité.

3 Objectif de remplacement

Une des plus grandes difficultés du problème de tournées de véhicules avec gestion des stocks est d'être capable de prendre des décisions court terme tout en assurant des améliorations long terme. Se focaliser sur l'optimisation du ratio logistique RL sur le court-terme ne conduit pas forcément à des solutions long-terme optimales. Soit un client situé très loin qui n'a pas d'assèchement prévu pour les prochains jours : une bonne décision court terme est d'éviter de le livrer afin de ne pas pénaliser le ratio coût logistique. Cependant, cette décision peut entraîner un surcoût très important sur le long terme. En effet si des ressources sont disponibles aujourd'hui pour effectuer une tournée quasi-optimale pour ce client, ce ne sera pas nécessairement le cas demain. Cet exemple illustre la règle suivante qui est le but à court terme : "ne jamais remettre à plus tard ce qu'on peut faire optimalement aujourd'hui".

Ce manque d'anticipation quand on minimise directement le ratio logistique sur le court terme a motivé l'introduction d'une fonction-objectif de substitution. Ainsi, le but à court terme est de minimiser le coût global supplémentaire par unité de produit livré, comparé au ratio logistique optimal RL^*. On note $RL^*(c)$ le ratio logistique optimal pour livrer un client c et aussi par :

$$coutT^*(t) = \sum_{c \in Clients(t)} RL^*(p) \times quantity(c) \quad (3)$$

le coût optimal d'une tournée t selon les quantités livrées à chaque client sur t. Ainsi, le ratio logistique modifié RL' est défini comme :

$$RL' = \frac{\sum_t (coutT(s) - coutT^*(t))}{QT} \quad (4)$$

Cependant, ceci nécessite de résoudre un autre problème : celui du calcul des bornes $RL^*(c)$ pour chaque client c (et donc du ratio logistique global RL^*). La tournée la plus courte desservant un client est un aller-retour chez ce client (du fait de l'inégalité triangulaire) et la quantité maximale livrable est la taille

du plus grand camion susceptible de le visiter. Cette observation définit une borne simple qui est tout à fait appropriée à l'objectif de substitution introduit ci-dessus. Ce calcul se fait dans une étape préliminaire dont la complexité est en temps $O(N\ M)$ avec N le nombre de clients et M le nombre de remorques.

4 Algorithme de recherche locale

De par sa complexité et sa taille, ce problème est typique des grands problèmes industriels d'optimisation combinatoire. Nous décrivons ici une heuristique à base de recherche locale pour le résoudre de manière effective et efficace. L'algorithme présenté ici fournit une solution de qualité en un temps très court (moins de 5 minutes). De plus, notre algorithme répond à certaines exigences en matière d'ingénierie logicielle : fiabilité, robustesse, portabilité, maintenabilité. Reprenant les propositions introduites dans les derniers travaux en matière de recherche locale haute performance (Estellon et al., 2008, 2009), la méthode de conception comporte 3 étapes : mise en place d'une stratégie de recherche, création de mouvements adaptés à la structure de notre problème et implémentation des algorithmes. Avant de présenter l'algorithme et de fournir des détails concernant les mouvements, nous introduirons l'algorithme glouton permettant d'obtenir une solution initiale.

4.1 Algorithme d'initialisation

L'algorithme glouton d'initialisation mis en place est basé sur l'urgence à livrer un client. Il consiste à ordonner les commandes et les assèchements par date croissante. Par assèchement, on entend première date à partir de laquelle le client est sous son seuil de sécurité. Les clients sont considérés un par un, triés par urgence. Pour chacun, on tente de réaliser l'insertion la moins coûteuse à la fin d'une tournée déjà existante. On compare ce coût à celui de la création d'une nouvelle tournée dédiée pour servir ce client avant cette date. Dans ce cas, on cherche une base et une source ouverte suffisamment proches pour créer une tournée élémentaire consistant à quitter la base, visiter la source et ce client puis revenir à la base, en utilisant un triplet de ressources disponibles. Des coupes permettent de trouver rapidement une solution parmi les ressources disponibles et les sites accessibles. De par sa définition, cet algorithme glouton ne revient jamais ni sur les décisions prises sur les dates et ni sur les volumes. Une fois cette première solution obtenue (en quelques secondes pour toutes les instances), on peut commencer la descente par recherche locale en utilisant une série de mouvements que nous allons détailler par la suite. A noter que cette solution est bien meilleure que la solution vide et des tests ont démontré qu'il est une étape indispensable pour atteindre des solutions de qualité car il accélère la convergence.

4.2 Stratégie de recherche

La stratégie de recherche est ici de type "first improvement descent" avec choix stochastique des mouvements (mais pas forcément uniformes). Nous n'utilisons pas ici de métaheuristiques avec de nombreux paramètres difficiles à régler. On retrouve ici trois phases correspondant à chacun des objectifs : la recherche locale se focalise d'abord sur le service des commandes non satisfaites en cherchant à minimiser les coûts associés. On cible donc les mouvements en adéquation avec cet objectif. Puis, on cherche à faire baisser le coût global des assèchements, en sélectionnant des mouvements performants pour ce sous-objectif. Enfin, une fois le nombre de commandes insatisfaites et le volume en assèchement nuls ou si on a atteint la limite de temps imparti à chacun de ces sous-objectifs, on consacre la recherche locale à la réduction des coûts logistiques. Pour cela, on active une gamme très large de mouvements variés sur les tournées, les opérations et les sites. A chaque étape, on accepte une transformation si elle ne dégrade ni le sous-objectif courant, ni les précédents.

Nous allons détailler dans la section suivante les familles de mouvements utilisés puis ensuite présenter deux sous-problèmes qui se révèlent cruciaux pour assurer une bonne performance globale de l'algorithme. Premièrement, le calcul pour une tournée, des dates de départ et de visites de chaque site et d'arrivée. Deuxièmement, le calcul des volumes à prendre aux sources et à livrer aux clients.

4.3 Les mouvements

Ce paragraphe a pour but de présenter les différents mouvements implémentés dans cet algorithme. Ces transformations peuvent dégrader, égaler ou améliorer un ou plusieurs sous-objectifs de la fonction objectif. La recherche locale cherche à accélérer le processus d'évaluation des mouvements. Chacun de ces mouvements nécessite un re-calcul des dates en conservant la date de l'opération concernée par la transformation et en mettant à jour soient toutes les dates des opérations précédentes jusqu'au début de la tournée, soient celles des opérations suivantes jusqu'à la fin.

Les transformations sont classées dans deux catégories : celles qui travaillent sur les opérations et celles qui agissent sur les tournées. Il est important de définir un pool de transformations diverses et variées (induisant des voisinages disjoints) afin de diversifier la recherche et aussi d'atteindre des solutions de meilleure qualité. Il est important aussi de spécialiser les transformations en fonction des spécificités du problème (car les choix aléatoires ne sont pas toujours appropriés pour toutes les situations) ; on intensifie ainsi la recherche et on accélère la convergence de l'heuristique. En résumé, on a donc des transformations qui

sont orthogonales, c'est à dire qu'elles diversifient et assurent la convergence vers des solutions de grandes qualités et qui sont aussi spécifiques afin d'intensifier la recherche et d'accélérer la convergence.

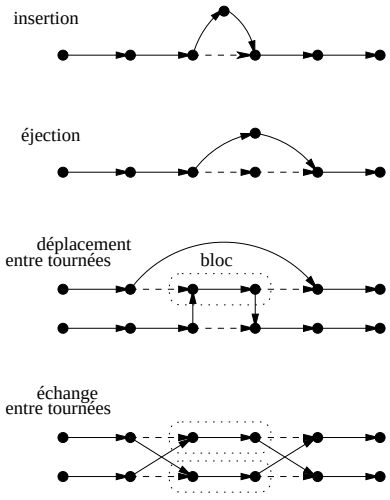

FIGURE 1 – Transformations sur les opérations (1).

Les transformations sur les opérations appartiennent aux types suivants : insertion, suppression, éjection, déplacement et échange (Figure 1 et Figure 2). Deux types d'insertion sont définis : le premier type consiste à insérer une opération dans une tournée existante. L'autre consiste à insérer un chargement suivi d'une livraison à l'intérieur d'une tournée (l'usine est choisie proche du client inséré). La suppression consiste à retirer un bloc d'opérations consécutives d'une tournée. Une éjection consiste à remplacer une opération existante par une nouvelle portant sur un autre site. Le déplacement cherche à extraire un bloc d'opérations d'une tournée et à la réinsérer à une autre position. Deux types de déplacements sont créés : déplacer des opérations d'une tournée à une autre, ou déplacer des opérations au sein d'une tournée. Un échange permet d'intervertir deux blocs différents d'opérations. Comme pour les déplacements, plusieurs types échanges sont définis : les déplacements de blocs entre tournées, les déplacements de blocs à l'intérieur d'une même tournée et aussi le miroir consistant en une inversion chronologique d'un bloc d'opérations au sein d'une tournée. Cette amélioration correspond au 2-opt, bien connu dans la résolution du problème de voyageur de commerce (Aarts et al. 1997).

Les transformations sur les tournées sont de types suivants : insertion, suppression, décalage, déplacement et échange. Comme pour les opérations, deux types d'insertion existent : l'insertion d'une tournée d'une opération ou l'insertion d'une tournée avec un chargement suivi d'une livraison. La suppression consiste à retirer une tournée existante. Le décalage translate une tournée dans le temps. Le déplacement cherche à extraire une tournée du planning de ses ressources

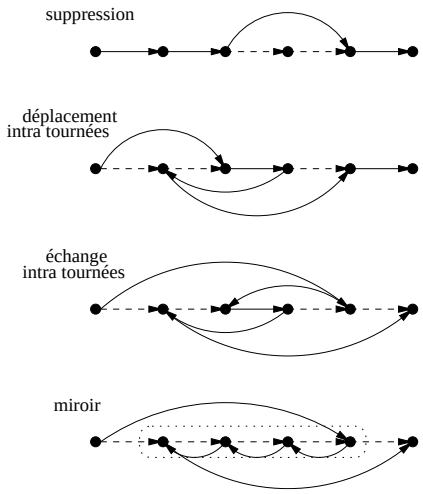

FIGURE 2 – Transformations sur les opérations (2).

et à la ré-insérer dans le planning d'autres ressources (on autorise ici à changer certaines ressources de la tournée et la date de début). L'échange est défini similairement : les ressources des tournées sont échangées et leur date de début peuvent être changées dans le temps. La fusion de deux tournées en une seule tout comme la séparation d'une tournée en deux sont aussi des transformations possibles.

Toutes ces transformations sont déclinées suivant différentes stratégies. La première option concerne la taille maximale des blocs pour les transformations où les blocs d'opérations sont utilisés. Ainsi, des transformations plus génériques sont définies afin d'augmenter la diversification si nécessaire : la (k, l)-éjection qui permet de remplacer k opérations existantes par l nouvelles sur des sites différents, le k-déplacement qui permet de bouger un bloc de k opérations, le (k, l)-échange consistant à échanger un bloc de k opérations par un bloc de l autres et enfin le k-miroir qui inverse un bloc de k opérations. Ensuite, la deuxième option permet de spécialiser les transformations en fonction de la phase d'optimisation. Par exemple, l'insertion d'une livraison à un client qui n'a pas de commande non satisfaite n'est pas intéressante quand on est en train de minimiser les coûts associés. De même pour les assèchements. Dans la même idée, échanger deux opérations entre deux sites très éloignés a peu de chance de solder par un succès quand on est en phase de minimisation des coûts logistiques. Plusieurs déclinaisons ont été mises en place qui diffèrent légèrement d'une transformation à l'autre : celles permettant un choix de position favorisant les livraisons satisfaisant une commande, celles favorisant la résolution des assèchements et enfin celles permettant de choisir des clients proches à insérer ou à échange afin de baisser les coûts logistiques. Puis, la troisième option concerne la direction du calcul des dates pour les tournées modifiées : on peut soit calculer les dates

vers l'avant en considérant que la date de début est fixée et que les dates des opérations suivantes se déduisent de celle-ci, soit vers l'arrière en fixant la date de fin et en remontant dans le temps. Cette option est valable pour toutes les transformations, sauf pour la suppression des tournées. Enfin, la quatrième et dernière option permet d'augmenter le nombre d'opérations dont la quantité sera modifiée par la routine d'affectation du volume. L'opération de re-calcul des volumes après une transformation peut permettre de résoudre des assèchements mais augmente considérablement le temps nécessaire à l'évaluation de la transformation.

Les voisinages explorés ici ont une taille d'environ $O(n^2)$ avec n le nombre d'opérations et de tournées dans la solution courante, mais la constante cachée dans la notation O est grande. Le nombre de transformations pour les trois phases *CO*, *CA* puis *RL* sont respectivement de 47, 49 et 71. Pour chaque phase, la transformation à appliquer est choisie aléatoirement dans chaque ensemble de transformations. Les distributions non-uniformes n'apportent pas d'amélioration significative et ne sont donc pas utilisées ici.

4.4 Algorithmique et implémentation

Cette section a pour but de présenter des détails concernant deux sous-routines du moteur de calcul, appelées à chaque évaluation des mouvements puis lors de chaque validation, sur la tournée modifiée ainsi que sur toutes celles impactées. La première fonction d'ordonnancement d'une tournée a pour but de recalculer les dates d'une tournée depuis une opération, jusqu'au début ou à la fin de la tournée (nous parlerons alors d'ordonnancement en avant ou ordonnancement en arrière). La seconde permet l'affectation des volumes en calculant les volumes livrés (resp. chargés) à chaque client (resp. source) d'une tournée, étant donné le niveau initial du camion.

4.4.1 Ordonnancement d'une tournée

Soit une tournée définie par une base, un ensemble ordonné d'opérations effectuées chacune sur des sources ou des clients. L'objectif de ce sous-problème est de visiter dans l'ordre indiqué chaque site (client ou source) d'une tournée, dans sa fenêtre de temps, en respectant l'ensemble des contraintes définies dans la section précédente et en autre, de ne pas dépasser les temps maximum de travail et de conduite. Le triplet de ressources réalisant la tournée est fixé et on peut donc en déduire les temps de trajet entre chaque point. On cherche ici à arriver au plus tôt, tout en minimisant les temps de pause au cours de la tournée. Certaines dates de visites peuvent être fixées au préalable : par exemple, on peut souhaiter recalculer uniquement les dates des opérations entre i et j avec $1 <= i < j <= k$ où k est le nombre d'opérations de la tournée. La date de visite de l'opération $(k-3)$ sera alors considérée comme étant la date de référence et le calcul pourra s'orienter, au choix, vers les visites futures ou passées. De plus, des temps d'attente additionnels peuvent apparaître. Si la date d'arrivée calculée pour un site est en dehors de ses fenêtres d'accès, un temps d'attente sera ajouté afin d'attendre l'ouverture. S'il existe entre deux sites s_1 et s_2 une pause due à un dépassement, tout éventuel temps d'attente avant s_2 sera intégré à cette pause et donc non décompté du temps de travail.

L'objectif de cet algorithme approché est de minimiser les temps improductifs, c'est-à-dire les pauses et les temps d'attente devant les grilles des sites. Cet algorithme inexact est linéaire en le nombre d'opérations. Bien qu'il ne fournisse pas de solutions optimales, il donne en pratique de très bons résultats. Des structures incrémentales permettent d'accéder pour chaque pas de temps et pour chaque site à sa prochaine fenêtre de temps d'ouverture. Pour chaque ressource, on accède aussi de la même manière à sa prochaine fenêtre de disponibilité. Chaque itération de la boucle étant réalisé en temps et en espace $O(1)$, l'algorithme complet s'exécute donc en temps $O(k)$ avec k le nombre de sites à visiter.

4.4.2 Affectation des volumes

Une fois les dates mises à jour sur les tournées impactées, il faut mettre à jour les volumes des opérations associées. Les dates étant désormais fixées, le problème consiste uniquement à décider de l'affectation des volumes de manière à respecter les contraintes d'inventaire tout en maximisant la quantité totale livrée à l'échelle de toutes les tournées. D'un point de vue théorique, ce problème peut se formuler simplement puisqu'il est polynomial : on peut en effet l'écrire comme un problème de flot maximum dans un réseau acyclique orienté. Le problème du calcul des volumes peut être résolu en temps $O(n^3)$ en utilisant un algorithme classique de flot maximum, avec n le nombre d'opérations. Cependant, une telle complexité n'est pas acceptable ici, même si cela garantit l'optimalité.

Ainsi, ce besoin d'efficacité algorithmique a motivé les développement d'un algorithme approché en temps $O(n \log n)$ pour résoudre efficacement ce problème d'affectation des dates. L'idée principale sous-jacente est simple : on affecte le maximum de quantité aux opérations triées par ordre chronologique. Cet ordre est crucial pour garantir le respect des contraintes relatives aux inventaires (conservation des flots et contraintes de capacité). En théorie des graphes, cet algorithme revient à pousser un maximum de flot dans un réseau acyclique orienté en respectant l'ordre topologique des noeuds.

Cependant, le réseau est ici très grand et n peut devenir tellement grand qu'il est nécessaire de trouver un compromis entre la complexité en temps (même linéaire) et la qualité de l'affectation des volumes. Dans le pire des cas en pratique, on peut imaginer avoir jusqu'à 1500 clients qui doivent être livrés deux fois par jour pendant la durée du planning, ce qui nous amène à $n = 45000$. Afin d'introduire un peu de flexibilité ici, l'algorithme a été conçu de manière à faire des affectations partielles, entre une affectation *minimale* et *complète*. Une affectation minimale consisterait à changer uniquement les volumes des opérations impactés (celles dont la date de début a été modifiée par la transformation courante) alors que l'affectation complète ré-affecte les volumes de toutes les opérations. Le but est donc de marquer comme impactées des opérations supplémentaires afin d'étendre la ré-affectation des volumes. En effet, changer la quantité livrée d'une opération est un processus délicat car augmenter (resp. diminuer) la quantité livrée (resp. chargée) peut entraîner un dépassement de capacité (resp. une pénurie) d'une opération future. La détermination du volume maximale livrable (ou chargeable) n'est donc pas un calcul trivial et direct.

Pour chaque site p, on note $\bar{n}_p$ le nombre d'opérations entre la première impactée et la dernière avant la fin du planning (les opérations sont dans l'ordre chronologique). Si aucune opération n'est impactée sur le site p, on a $\bar{n}_p = 0$. On définit le nombre total d'opérations impactées $\bar{n} = \sum_p \bar{n}_p$. Quand l'ensemble des opérations impactées ne contient que des opérations dont la date a été modifiée par la transformation, on observe en pratique que $\bar{n} \ll n$, car chaque transformation modifie au mieux deux tournées (le nombre de sites visités par une tournée est généralement petit). Par conséquence, il est plus intéressant de fournir un algorithme qui tourne en un temps linéaire en $O(\bar{n})$, et pas seulement en $O(n)$. Avant de présenter l'algorithme en lui-même, nous allons préciser comment sont calculées les quantités maximales livrables (les quantités chargeables se déduisant de manière symétrique). On note $niveauC(c, o)$ (resp. $niveauR(r, o)$) le niveau du client c (resp. de la remorque r) avant le début de l'opération o et par $eviterDebord(c, o)$ la quantité maximale qui peut être livrée à un client c pendant une opération o sans provoquer de débordement jusqu'à la fin du planning. Ainsi, la quantité livrable à une opération o, notée $livrable(o)$, est bornée supérieurement par $\min\{niveauR(r, o), eviterDebord(c, o)\}$. Cette borne est renforcée car la quantité restante dans la remorque après une livraison doit être suffisante pour éviter les assèchements de clients visités jusqu'au prochain chargement. On note $eviterAssech(c, o)$, la quantité minimale à livrer à une opération o pour éviter un assèchement avant la fin du planning. La quantité minimale $necessaire(r, o)$ qui doit rester dans la remorque r après une opération o pour éviter un assèchement ensuite est calculée en sommant $eviterAssech(c, o)$ pour toutes les opérations entre la courante et le prochain chargement. On a donc :

$$livrable(o) \leq \min\{niveauR(r, o) - necessaire(r, o),\ eviterDebord(c, o)\} \quad (5)$$

Connaissant la liste d'opérations modifiées ordonnées chronologiquement pour chaque remorque, client et usine, toutes les structures de données mentionnées ci-dessus sont calculables en $O(\bar{n})$ en temps. La mise à jour de $niveauC(c, o)$ (resp. $niveauR(r, o)$) pour chaque opération o est réalisée en balayant en avant les opérations livrant le client c (resp. réalisées par la remorque r). Celle de $eviterDebord(c, o)$ et de $eviterAssech(c, o)$ pour chaque opération o est fait en balayant en arrière les opérations du client c (on stocke pour cela les consommations cumulées jusqu'à la fin du planning). Enfin, calculer $necessaire(r, o)$ pour chaque opération o se fait en balayant en arrière les opérations des tournées réalisées par la remorque r. L'ordonnancement de l'ensemble des opérations impactées est en $O(\bar{n} \log \bar{n})$ en temps dans le pire des cas. Toute cette affectation de volume est donc fait en temps linéaire, puisque le calcul des quantités maximales livrables / chargeables nécessite un temps constant grâce à l'utilisation de structures de données adéquates. L'algorithme global a donc une complexité de $O(\bar{n} \log \bar{n})$ en temps. Des tests sur les instances (lorsque $CA = 0$) ont démontré que cet algorithme était 100 fois plus rapide que l'algorithme glouton global et 2000 fois plus rapide que l'algorithme de flot optimal, tout en restant à moins de 2 % de l'optimal.

Finalement, une fois ces volumes affectés, le calcul du gain d'une transformation peut se faire efficacement : la variation du coût des tournées est fait pendant la phase d'ordonnancement et celle du volume livré est obtenue pendant la phase d'affectation des volumes, sans ajout de complexité. On déduit aussi les variations du nombre d'assèchements ou de commandes non satisfaites pendant l'affectation. Ce dernier calcul nécessite par contre une boucle en $O(\log H)$ en temps, avec H le nombre de pas de temps jusqu'à l'horizon final (on recherche le premier pas de temps sous le seuil de sécurité).

4.4.3 Détails d'implémentation

Tous les ensembles (ordonnés ou non, fixe ou dynamique) utilisés au cours de cette recherche locale sont implémentés sous la forme de tableaux, afin d'améliorer la gestion de la mémoire cache. Les allocations de mémoires sont évitées autant que possible : toutes les structures de données sont allouées au démarrage de la recherche locale. Nous utilisons des piles d'objets afin d'éviter la création de nouveaux objets "tournée"

ou "opération" au cours de la recherche locale. Afin d'améliorer les routines d'acceptation et de retour arrière permettant respectivement de valider une transformation ou de revenir à l'état précédent après l'évaluation d'une transformation, toutes les variables de décisions sont dupliquées en des champs temporaires. Ainsi, un retour arrière consiste uniquement en une recopie des champs stockés. Pour optimiser les suppressions/insertions d'objets dans les listes, on utilise ici des listes doublement chaînées mises à jour elles-aussi en temps $O(1)$. Enfin, pour garantir l'efficacité des algorithmes, nous faisons appel à des structures de données stockant pour chaque opération, son successeur/prédécesseur dans la tournée et pour cette ressource (ces listes sont mises à jour uniquement lors d'une acceptation).

5 Résultats

L'algorithme complet de recherche locale a été implémenté en C# 2.0 (pour une exécution dans l'environnement Microsoft .NET 2.0). Le programme compte 30000 lignes de code, dont 6000 dédiées à la vérification de la validité des structures de données incrémentales, à chaque itération de l'algorithme (en mode *debug*). Toutes les statistiques et les résultats présentés ici ont été obtenus (sans parallélisation) sur un ordinateur équipé du système d'exploitation Windows Vista 64 bits et d'un processeur Intel Xeon X5365 (CPU 3 GHz, cache L1 64 Ko, cache L2 4 Mo, RAM 8 Go).

L'algorithme de recherche locale réalise près de 50000 mouvements par seconde, même pour des instances de grande taille (plus de mille sites et une centaine de ressources). Notre algorithme visite alors plus d'un million de solutions de l'espace de recherche par minute. Quand on travaille sur un horizon de 15 jours, la mémoire allouée par le programme ne dépasse pas les 30 Mo pour des instances de taille moyenne (une centaines de sites) et 150 Mo pour les instances plus grande (mille sites). A noter que le temps d'exécution de l'algorithme glouton d'initialisation est de l'ordre de quelques secondes pour les instances les plus grandes. Le taux d'acceptation des mouvements varie entre 1% et 10% suivant les instances et les phases d'optimisation. Ce taux, quasiment constant au long de la recherche, garantit une bonne diversification sans usage de métaheuristique.

Notre algorithme a été testé sur des instances court terme (15 jours) avec différentes caractéristiques : réalistes (provenant de conditions opérationnelle), pathologiques (par exemple, les usines s'arrêtent plusieurs jours), très grandes (avec 1500 sites et plus de 300 ressources). Le tableau Table 1 présente une partie des résultats obtenus sur ces jeux de données. Il n'y a aucune commande non satisfaite ni aucun assèchement dans ces jeux de données. Les 2 colonnes *gain RL'* et *gain RL* présentent respectivement les gains en ratio logistique modifié (avec la fonction objectif de substitution) et en ratio logistique non modifié (en pourcentages par rapport à l'algorithme glouton).

data	$gainRL'$	$gainRL$
A01	37.3 %	13.7 %
A02	50.1 %	26.5 %
A03	28.5 %	14.0 %
A04	51.1 %	21.7 %
A05	65.6 %	30.5 %
A06	33.5 %	26.1 %
A07	47.7 %	20.5 %
A08	51.9 %	29.5 %
A09	54.5 %	28.0 %
A10	49.8 %	26.9 %
A11	51.2 %	23.7 %
A12	33.4 %	13.5 %
A13	41.1 %	19.8 %
A14	40.7 %	14.8 %
A15	36.8 %	15.6 %
A16	49.7 %	28.0 %
A17	48.7 %	30.9 %

TABLE 1 – Résultats court terme (extrait).

Sur les instances à long terme, on mesure l'apport de la fonction objectif modifiée. On présente ici 5 jeux de données réels sur 105 jours, pour lesquels on a laissé tourner l'algorithme 5 minutes par tranche de 15 jours. Pour chaque tranche, on ne conserve que les tournées débutant le premier jour, on les fixe en mettant à jour les inventaires, puis on itère au jour suivant. Dans les tableaux Table 2 et Table 3, on retrouve le descriptif des instances long terme avec le nombre de clients, de sources, de dépôts, de clients n'acceptant pas les livraisons en dehors des commandes (*Invent.*), des commandes et le nombre de ressources (conducteurs, tracteurs, remorques). Comme la recherche locale est stochastique, cinq exécutions ont été réalisées avec une graine différente pour chaque lancement. La colonne "Moy 1h" (resp "Moy 1m" et "Moy 5m") du tableau Table 4 indique les gains obtenus sur le ratio logistique par une recherche locale limitée à 1 heure (resp à 1 et 5 minutes) de calcul pour chaque sous-planning de 15 jours, par rapport à la solution obtenue par l'algorithme glouton. La colonne "Pire 1m" présente le moins bon résultat obtenu sur les cinq lancements en 1 minute. Le gain sur le ratio logistique obtenus par recherche locale en 5 minutes de calcul pour chaque sous planning de 15 jours, comparé à la solution trouvée par le glouton ou par les experts logistiques, dépasse les 20 % en moyenne pour ces cinq instances à long terme. Sur ces instances, la fonction objectif de substitution apporte un gain de plus de 5 % en moyenne, par rapport à la fonction objectif classique.

Inst	Clients	Usines	Dépôts	Invent.
L1	75	6	1	19
L2	75	6	1	20
L3	175	8	1	36
L4	165	4	1	33
L5	198	8	7	3
moyenne	138	6	2	101

TABLE 2 – Instances long terme (1).

Inst	Conduct.	Tract.	Rem.	Command.
L1	35	21	5	56
L2	35	21	5	55
L3	35	21	12	189
L4	24	11	7	167
L5	12	12	12	40
moyenne	28	17	8	101

TABLE 3 – Instances long terme (2).

Inst	Pire 1m	Moy 1m	Moy 5m	Moy 1h
L1	23.8 %	24.6 %	26.3 %	26.5 %
L2	22.3 %	23.5 %	24.9 %	25.2 %
L3	5.2 %	5.8 %	8.3 %	11.2 %
L4	9.9 %	11.2 %	14.0 %	18.9 %
L5	32.5 %	34.2 %	35.7 %	35.9 %
moyenne	18.7 %	19.9 %	21.8 %	23.5 %

TABLE 4 – Résultats long terme.

6 Conclusion

Après avoir introduit un problème réel d'optimisation de tournées de véhicules avec gestion des stocks, nous avons présenté deux contributions dans ce papier. Tout d'abord, une fonction objectif basée sur des bornes inférieures est mise en place pour assurer une optimisation sur le long terme en construisant des plannings sur le court terme, apportant un gain de plus de 5 % par rapport à la fonction objectif classique. Ensuite, une heuristique recherche locale pure et directe est décrite pour résoudre efficacement, en quelques minutes, le problème réel sur du court terme (15 jours), même sur des instances réelles de très grande taille (consommations et productions non linéaires, plusieurs centaines de sites, plusieurs dizaines de ressources). Pour éviter une décomposition du problème, nous nous appuyons notamment sur une heuristique cruciale pour résoudre le problème d'affectation des volumes des tournées, 2000 fois plus rapide que l'algorithme de flot optimal, tout en restant à moins de 2 % de l'optimal. Des structures de données incrémentales permettent de garantir l'efficacité des évaluations de mouvements. Des tests sur de nombreuses instances de différentes natures ont permis de démontrer que cet algorithme permet des réductions de coûts dépassant les 20 % en moyenne en comparaison à des solutions calculées par des planificateurs experts et même par rapport à notre heuristique gloutonne. Ce programme est désormais utilisé quotidiennement en production. Les perspectives d'évolution concerne notamment l'introduction de mouvements à voisinage large, l'ajout de nouvelles contraintes opérationnelles et l'amélioration des bornes inférieures utilisées dans la fonction objectif modifiée.

Références

Aarts E., J. Lenstra, 1997. Local Search in Combinatorial Optimization. *Wiley-Interscience Series in Discrete Mathematics and Optimization*, UK.

Bell W. , L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, P. Prutzman, 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13(6), pp. 4–23.

Campbell A., L. Clarke, A. Kleywegt, M. Savelsbergh, 1998. The inventory routing problem. *Fleet Management and Logistics*, pp. 95–113. Kluwer Academic Publishers.

Campbell A., L. Clarke, M. Savelsbergh, 2002. Inventory routing in practice. *The Vehicle Routing Problem*, pp. 309–330. SIAM Monographs on Discrete Mathematics and Applications 9. SIAM.

Campbell A., M. Savelsbergh, 2004a. A decomposition approach for the inventory-routing problem. *Transportation Science* 38(4), pp. 488-502.

Campbell A., M. Savelsbergh, 2004b. Delivery volume optimization. *Transportation Science* 38(2), pp. 210-223.

Campbell A., M. Savelsbergh, 2004c. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science* 38(3), pp. 369–378.

Estellon B., F. Gardi, K. Nouioua, 2008. Two local search approaches for solving real-life car sequencing problems. *European Journal of Operational Research* 191(3), pp. 928–944.

Estellon B., F. Gardi, K. Nouioua, 2009. High-performance local search for task scheduling with human resource allocation. *In Proceedings of SLS 2009*, LNCS 5752, pp. 1-15, Springer-Verlag.

Lau H., Liu, H. Ono, 2002. Integrating local search and network flow to solve the inventory routing problem. *Proceedings of AAAI 2002, the 18th National Conference on Artificial Intelligence*, pp. 9–14. AAAI Press.

Savelsbergh M., J.-H. Song, 2007. Inventory routing with continuous moves. *Computers and Operations Research* 34(6), pp. 1744–1763.

Savelsbergh M., J.-H. Song, 2008. An optimization algorithm for the inventory routing with continuous moves. *Computers and Operations Research* 35(7), pp. 2266–2282.