Université de Montréal

Méthodes Exactes et Heuristiques pour le Problème de Tournées avec Fenêtres de Temps et Réutilisation de Véhicules

par

Nabila Azi

Département d'informatique et de recherche opérationnelle

Faculté des Arts et des Sciences

Thèse présentée à la Faculté des arts et des Scienses

en vue de l'obtention du grade de

Docteur ès sciences (Ph.D.)

en informatique

Août 2010

Université de Montréal

Faculté des arts et des Scienses

Cette thèse intitulée:

Méthodes Exactes et Heuristiques pour le Problème de Tournées avec Fenêtres de Temps et Réutilisation de Véhicules

présentée par:

Nabila Azi

a été évaluée par un jury composé des personnes suivantes:

Jacques Ferland

(président-rapporteur)

Michel Gendreau

(directeur de recherche)

Jean-Yves Potvin

(co-directeur)

Louis-Martin Rousseau

(membre du jury)

Rodolphe Giroudeau

(examinateur externe)

Christian Léger

(représentant du doyen de la F.A.S.)

Résumé

Cette thèse porte sur les problèmes de tournées de véhicules avec fenêtres de temps où un gain est associé à chaque client et où l'objectif est de maximiser la somme des gains recueillis moins les coûts de transport. De plus, un même véhicule peut effectuer plusieurs tournées durant l'horizon de planification. Ce problème a été relativement peu étudié en dépit de son importance en pratique. Par exemple, dans le domaine de la livraison de denrées périssables, plusieurs tournées de courte durée doivent être combinées afin de former des journées complètes de travail. Nous croyons que ce type de problèmes aura une importance de plus en plus grande dans le futur avec l'avènement du commerce électronique, comme les épiceries électroniques, où les clients peuvent commander des produits par internet pour la livraison à domicile.

Dans le premier chapitre de cette thèse, nous présentons d'abord une revue de la littérature consacrée aux problèmes de tournées de véhicules avec gains ainsi qu'aux problèmes permettant une réutilisation des véhicules. Nous présentons les méthodologies générales adoptées pour les résoudre, soit les méthodes exactes, les méthodes heuristiques et les méta-heuristiques. Nous discutons enfin des problèmes de tournées dynamiques où certaines données sur le problème ne sont pas connues à l'avance.

Dans le second chapitre, nous décrivons un algorithme exact pour résoudre

un problème de tournées avec fenêtres de temps et réutilisation de véhicules où l'objectif premier est de maximiser le nombre de clients desservis. Pour ce faire, le problème est modélisé comme un problème de tournées avec gains. L'algorithme exact est basé sur une méthode de génération de colonnes couplée avec un algorithme de plus court chemin élémentaire avec contraintes de ressources.

Pour résoudre des instances de taille réaliste dans des temps de calcul raisonnables, une approche de résolution de nature heuristique est requise. Le troisième chapitre propose donc une méthode de recherche adaptative à grand voisinage qui exploite les différents niveaux hiérarchiques du problème (soit les journées complètes de travail des véhicules, les routes qui composent ces journées et les clients qui composent les routes).

Dans le quatrième chapitre, qui traite du cas dynamique, une stratégie d'acceptation et de refus des nouvelles requêtes de service est proposée, basée sur une anticipation des requêtes à venir. L'approche repose sur la génération de scénarios pour différentes réalisations possibles des requêtes futures. Le coût d'opportunité de servir une nouvelle requête est basé sur une évaluation des scénarios avec et sans cette nouvelle requête.

Enfin, le dernier chapitre résume les contributions de cette thèse et propose quelques avenues de recherche future.

Mots clés. Problèmes de tournées avec gains, fenêtres de temps, réutilisation de véhicules, génération de colonnes, chemin plus court élémentaire avec contraintes de ressources, recherche adaptative à grand voisinage, stratégie d'acceptation et de refus dynamique.

Abstract

This thesis studies vehicle routing problems with time windows, where a gain is associated with each customer and where the objective is to maximize the total gain collected minus the routing costs. Furthermore. the same vehicle might be assigned to different routes during the planning horizon. This problem has received little attention in the literature in spite of its importance in practice. For example, in the home delivery of perishable goods (like food), routes of short duration must be combined to form complete workdays. We believe that this type of problem will become increasingly important in the future with the advent of electronic services, like e-groceries, where customers can order goods through the Internet and get these goods delivered at home.

In the first chapter of this thesis, we present a review of vehicle routing problems with gains, as well as vehicle routing problems with multiple use of vehicles. We discuss the general classes of problem-solving approaches for these problems, namely, exact methods, heuristics and metaheuristics. We also introduce dynamic vehicle routing problems, where new information is revealed as the routes are executed.

In the second chapter, we describe an exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles, where the first objective is to maximize the number of served customers. To this end, the problem is

modeled as a vehicle routing problem with gains. The exact algorithm is based on column generation, coupled with an elementary shortest path algorithm with resource constraints.

To solve realistic instances in reasonable computation times, a heuristic approach is required. The third chapter proposes an adaptative large neighborhood search where the various hierarchical levels of the problem are exploited (i.e., complete vehicle workdays, routes within workdays and customers within routes).

The fourth chapter deals with the dynamic case. In this chapter, a strategy for accepting or rejecting new customer requests is proposed. This strategy is based on the generation of multiple scenarios for different realizations of the requests in the future. An opportunity cost for serving a new request is then computed, based on an evaluation of the scenarios with and without the new request.

Finally, the last chapter summarizes the contributions of this thesis and proposes future research avenues.

Keywords. Vehicle routing problems with gains, time windows, multiple use of vehicles, column generation, elementary shortest path with resource constraints, adaptive large neighborhood search, dynamic acceptation and rejection strategy.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Je désire avant tout exprimer ma profonde gratitude à mes directeurs, les professeurs Michel Gendreau et Jean-Yves Potvin, pour leurs encouragements et pour le soutien financier qu'ils m'ont accordé tout au long de cette thèse. Je voudrais remercier le professeur Gendreau pour m'avoir proposé ce sujet de recherche aussi intéressant et de m'avoir fait bénéficier de son expérience pertinente dans le domaine des tournées de véhicules. Je tiens aussi à remercier le professeur Potvin pour sa disponibilité et son support, ainsi que pour les remarques éclairées qu'il m'a prodiguées tout au long de cette recherche.
Je remercie également les professeurs Jacques Ferland, Louis-Martin Rousseau et Rodolphe Giroudeau d'avoir accepté de faire partie du jury.

Je profite de cette occasion pour remercier les professeurs, chercheurs, professionnels et étudiants du Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT), pour avoir su créer une ambiance agréable et pour les moyens de travail qu'ils ont mis à notre disposition. Je remercie particulièrement Serge Bisaillon pour son énorme aide dans l'environnement de Cplex et pour sa grande disponibilité durant mes problèmes de déboggage.

Finalement, tout mon respect va à ma mère pour ses encouragements et ce malgré l'éloignement, à mon mari pour sa patience durant ces années de travail et à mes trois enfants Mounir, Selma et Mohamed pour leur présence chaleureuse.

Je ne peux m'empêcher en ce moment d'avoir une pensée à mon défunt père qui avait un grand respect pour les études.

Chapitre 1

Introduction

1.1 Motivation et contexte

Dans un contexte économique toujours plus concurrentiel, l'optimisation des ressources prend une place très importante dans les prises de décision des entreprises, y compris celles évoluant dans le domaine des transports. Des efforts considérables sont donc entrepris pour réduire les coûts et la consommation des différentes ressources, qu'elles soient matérielles ou humaines.

Le problème de tournées de véhicules (VRP pour *Vehicle Routing Problem*) à moindre coût est justement un problème logistique important pour la distribution de biens et le transport de personnes. En effet, plusieurs applications réelles, telles le transport scolaire, le transport adapté et les services de courrier entraînent la confection de tournées de véhicules. Nous nous intéressons plus particulièrement dans cette thèse au problème de tournées avec fenêtres de temps et réutilisation de véhicules. C'est une extension du problème classique de tournées de véhicules avec fenêtres de temps où un intervalle de temps est associée à chaque client pour contraindre le temps de début de service. Ici, toutefois, un même véhicule effectue

plusieurs tournées différentes sur l'horizon de planification, dû à une contrainte sur la durée maximale de chaque tournée. Si l'on suppose que le nombre de véhicules est insuffisant pour desservir l'ensemble des clients, l'objectif est de visiter un sous-ensemble de clients qui maximise le gain total moins les coûts de transport. Le problème s'apparente alors à la version multi-véhicules d'un problème de tournée avec gains (voir la section suivante).

Il faut noter que les problèmes de tournées avec réutilisation de véhicules ont été rarement étudiés dans la littérature même si l'on retrouve de nombreuses applications pratiques. Ainsi, avec l'avènement du commerce électronique, on observe une prolifération des magasins en ligne qui offrent des services de livraison à domicile. Dépendemment du contexte, le client doit être présent au moment de la livraison (service attendu) ou non (service non attendu). Le commerce électronique est devenu incontournable dans des secteurs comme le voyage (transport, hôtels), les biens informatiques (logiciels, ordinateurs), les biens culturels (livres, musique), ainsi que le secteur des épiceries électroniques qui se caractérisent par la distribution de produits alimentaires périssables. Dans ce dernier cas, il est important d'effectuer plusieurs tournées de courte durée afin de préserver la qualité des produits livrés. De plus, le commerçant se doit de visiter les clients les plus profitables, soit ceux pour lesquels le revenu ou gain est important par rapport au coût de transport (e.g., si la commande est importante et la position géographique du client est avantageuse).

Dans ce chapitre, nous présentons un état de l'art sur le problème du voyageur de commerce (TSP pour *Traveling Salesman Problem*) avec gains et sa variante multi-véhicules. Dans ces problèmes, un gain est associé à chaque sommet et l'objectif est de trouver une tournée qui maximise le gain total, sujet à une borne supérieure sur le coût de la tournée.

Dans la section 1.2, nous définissons précisément ces problèmes et présentons

une formulation en nombres entiers pour le problème du voyageur de commerce dit sélectif. La section 1.3 présente ensuite les méthodes de résolution proposées dans la littérature, suivie de la section 1.4 qui s'intéresse au cas multi-véhicules. Les problèmes de tournées avec réutilisation des véhicules sont l'objet de la section 1.5, avec une emphase sur le problème de livraison à domicile de denrées périssables. Dans la section 1.6, on présente quelques travaux traitant du cas dynamique, où toutes les informations sur le problème ne sont pas connues à l'avance.

1.2 Problèmes de voyageur de commerce avec gains

Les trois classes de problèmes regroupés sous le vocable de problèmes de voyageur de commerce avec gains sont les suivantes [22] :

- Problème du voyageur de commerce sélectif (STSP pour *Selective TSP*) : l'objectif est de trouver une tournée qui maximise le gain total, sujet à une borne supérieure sur le coût de la tournée.
- Problème du voyageur de commerce avec collecte de prix (PCTSP pour *Prize Collecting TSP*) : l'objectif est de trouver une tournée de coût minimal, sujet à une borne inférieure sur le gain total de la tournée.
- Problème de tournée profitable (PTP pour *Profitable Tour Problem*) : l'objectif est de trouver une tournée qui optimise une combinaison linéaire des deux objectifs précédents.

Il faut noter que la notion de coût d'une tournée se réduit souvent à la durée ou à la longueur totale de la tournée, soit la somme des temps de parcours ou des longueurs des arcs qui composent cette tournée. Puisque le problème au-

quel nous nous intéressons se rapproche davantage du STSP et du PTP, nous nous concentrerons sur ces problèmes dans la suite, tout en incluant le problème d'orientation (OP pour *Orientering Problem*), qui est un problème bien connu et qui s'apparente au STSP.

La première description du STSP se trouve dans [31]. Les auteurs s'intéressent alors à un problème de tournées de véhicules périodique avec prise en compte de l'inventaire où une flotte de véhicules doit livrer du carburant à un nombre important de clients à chaque jour. Le problème est modélisé comme un STSP où le gain représente une mesure d'urgence.

Au début des années 80, on retrouve également une description d'un problème similaire, soit le problème d'orientation [74]. Ce problème tient son origine d'un sport qui se pratique à l'extérieur dans des régions montagneuses ou forestières. Des compétiteurs, munis d'une carte, doivent se rendre d'une origine à une destination données à l'intérieur d'un temps limite, tout en visitant des points de contrôle. Un score est associé à chacun des points de contrôle et le but est de maximiser le score total recueilli. Le STSP est en fait une forme particulière du OP, où l'origine et la destination se confondent. Quelques généralisations du OP ont aussi été abordées dans la littérature. Ainsi, Kantor et Rosenwein [38] ont traité le problème d'orientation avec fenêtres de temps, où les points de contrôle ne peuvent être visités qu'à l'intérieur d'un certain intervalle de temps. Également, Chao et al. [18] ont étendu ce problème au cas d'une équipe composée de plusieurs compétiteurs.

Le STSP, tout comme le OP et le PTP, appartiennent à la classe des problèmes NP-complets. En effet, il s'agit de généralisations du problème du voyageur de commerce qui est lui-même NP-complet. Nous donnons ci-dessous une formulation en nombres entiers de la version orientée du STSP. Soit $G = (N, A)$ un graphe complet, où $N = \{1, ..., n\}$ est l'ensemble des n sommets du graphe, avec

le sommet 1 jouant le rôle de dépôt, et où A correspond à l'ensemble des arcs. Associons un gain $g_i \geq 0$ à chaque sommet $i \in N$ (avec $g_1 = 0$) et un coût $c_{ij} \geq 0$ à chaque arc $(i, j) \in A$ (souvent, le coût correspond à la longueur de l'arc). Le STSP consiste à déterminer un circuit élémentaire de gain maximal contenant le sommet 1 et dont le coût ne dépasse pas une borne C_{max}. La formulation présentée dans [21] associe une variable binaire x_{ij} à chaque arc $(i, j) \in A$, où x_{ij} est égal à 1 si l'arc correspondant est utilisé dans la solution, et 0 sinon. On retrouve aussi une variable binaire y_i à chaque sommet $i \in N$, où y_i est égale à 1 si le sommet correspondant est visité, et 0 sinon.

En utilisant ces variables, le STSP peut se formuler comme le programme en nombres entiers suivant :

$$\text{Max} \sum_{i \in N} g_i y_i \tag{1.1}$$

$$\text{sujet à :} \quad \sum_{j \in N \setminus \{i\}} x_{ij} = y_i, \quad i \in N, \tag{1.2}$$

$$\sum_{i \in N \setminus \{j\}} x_{ij} = y_j, \quad j \in N, \tag{1.3}$$

$$\sum_{i \in S} \sum_{j \in S \setminus \{i\}} \leq |S| - 1 \quad (S \subseteq N \setminus \{1\}, 2 \leq |S| \leq n - 2), \tag{1.4}$$

$$\sum_{i \in N} \sum_{j \in N \setminus \{i\}} c_{ij} x_{ij} \leq C_{max}, \tag{1.5}$$

$$y_1 = 1, \tag{1.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in A, \tag{1.7}$$

$$y_i \in \{0, 1\} \qquad \forall\, i \in N. \tag{1.8}$$

Les contraintes (1.2) et (1.3) sont les contraintes d'affectation. Les contraintes (1.4) éliminent les sous-tours qui ne contiennent pas le sommet 1. Enfin, la contrainte (1.5) impose une contrainte de coût maximal.

1.3 Approches de résolution pour le STSP, l'OP et le PTP

Dans les sections qui suivent, nous présentons les principales heuristiques, métaheuristiques et méthodes exactes qui ont été proposées dans la littérature pour résoudre le STSP, l'OP et le PTP.

1.3.1 Heuristiques

Les heuristiques pour résoudre les problèmes avec gains doivent aborder deux niveaux de décision. D'abord, un sous-ensemble de sommets est sélectionné selon un certain critère (faisant intervenir, le plus souvent, le gain associé à un sommet). Ensuite, un parcours (tournée ou chemin) de moindre coût est déterminé pour desservir ces sommets. Ces deux niveaux de décision sont évidemment inter-reliés. Nous commençons par présenter les heuristiques simples qui ont été développées pour ces problèmes.

Dans [74], l'auteur présente deux heuristiques d'insertion pour le problème OP. La première heuristique est de nature stochastique, tandis que la deuxième est de nature déterministe. La première heuristique fonctionne ainsi. À chaque sommet i qui n'est pas encore dans le chemin partiel courant, on associe une mesure de désirabilité D_i donnée par :

$$D_i = \left(\frac{S_i}{c_{last,i}} \right)^r$$

où S_i est le score (gain) associé au sommet i, $c_{last,i}$ correspond au coût (distance) entre le dernier sommet du chemin courant et le sommet i et r est un paramètre dont la valeur est fixée à 4. Les k meilleurs sommets, selon cette mesure de désirabilité, sont retenus et une probabilité de sélection leur est associée, qui est

égale à :

$$P_i = \frac{D_i}{\sum_{j=1}^{k} D_j},\ i = 1, ..., k$$

Le sommet i est choisi selon cette loi de probabilité et est ensuite inséré dans le chemin partiel courant. Cette procédure d'insertion est répétée jusqu'à ce que l'on ne puisse plus insérer de sommets sans violer la borne de coût maximal C_{max}. L'auteur génère près de 3 000 routes différentes de cette façon et celle ayant le score le plus élevé est choisie.

La deuxième heuristique divise l'espace géographique en secteurs déterminés par deux cercles concentriques et un pivot. Les secteurs changent selon la longueur des rayons des deux cercles et l'angle du pivot (rotations de $\pi/2$ degrés). Le processus de construction d'une route dans un secteur donné s'arrête lorsque tous les sommets de ce secteur sont visités ou lorsqu'il n'est plus possible de visiter d'autres sommets sans violer la contrainte sur le coût maximal. Au total, 48 configurations différentes pour les secteurs sont examinées et la meilleure solution obtenue est retenue à la fin.

Les auteurs dans [32] présentent une heuristique qui se divise en trois étapes, soit : construction d'une route, puis amélioration et mise à jour du centre de gravité de la route. Une route est d'abord construite en associant une mesure W_i à chaque sommet i non encore visité dans la route partielle courante. Cette mesure repose sur trois critères pondérés par les paramètres α, β et γ, en l'occurence un score S_i, la distance au centre de gravité C_i et la somme des distances vers les deux foyers de l'ellipse E_i, formés de l'origine et de la destination. La mesure W_i est donnée par

$$W_i = \alpha S_i + \beta C_i + \gamma E_i,$$

avec $\alpha + \beta + \gamma = 1$. Les sommets ayant les plus grandes mesures sont sélectionnés et insérés à moindre coût tout en respectant la contrainte de coût maximal. Dans la seconde étape, une procédure d'amélioration $2-opt$ [46] est appliquée. Fina-

lement, la troisième étape met à jour le centre de gravité de la route. On répète cette procédure jusqu'à que l'on rencontre une solution déjà obtenue. La meilleure solution visitée est alors retournée.

Les auteurs dans [33] définissent une mesure qui comprend un élément d'apprentissage. Plus précisément, un poids est associé à chaque sommet i qui n'est pas dans le chemin courant, en l'occurrence :

$$W_i = \alpha SM_i + \beta C_i + \gamma E_i,$$

où $\alpha+\beta+\gamma = 1$. SM_i prend en compte la densité de gain à proximité du sommet i donnée par :

$$SM_i = \sum_j S_j e^{-\mu c_{ij}} * LM_i$$

où μ est un paramètre qui reflète l'éloignement des sommets et LM_i est une mesure d'apprentissage tenant compte de l'information recueillie sur les solutions déjà visitées, soit :

$$LM_i = \frac{1}{|R(i)|} \sum_{l \in R(i)} \frac{S(l)}{\sum_{k \in R(i)} S_k}$$

Dans cette équation, $R(i)$ correspond à l'ensemble des routes contenant le sommet i dans les solutions précédentes et $S(l)$ est le score total de la route l, soit la somme des scores des sommets qui font partie de cette route.

Une heuristique en deux phases est proposée dans [39] : d'abord la construction d'un chemin, suivie d'une amélioration. Dans la phase de construction, deux alternatives sont évaluées pour déterminer le prochain sommet à insérer dans la route partielle courante. La première approche est déterministe et calcule le ratio entre le revenu potentiel du sommet i et la distance entre le sommet i et le dernier sommet de la route courante. La seconde approche est de nature stochastique et choisit le prochain sommet à insérer aléatoirement. En phase d'amélioration, on essaie de remplacer un sommet dans le chemin courant par un autre sommet non visité qui permet d'augmenter le score ou gain total, tout en demeurant réalisable.

L'auteur propose aussi des remplacements de sous-groupes de sommets dans le but d'accroître le score total.

Dans [58], les auteurs s'intéressent à quatre versions du problème d'orientation : une version où l'origine et la destination se confondent (et qui correspond donc au STSP) ; une version où l'origine et la destination se confondent et où l'on permet de revisiter des sommets ; enfin, deux autres versions qui correspondent aux deux premières, mais où l'origine et la destination sont distinctes. Les auteurs appliquent une méthode en quatre phases qu'ils adaptent à chacune des versions. Dans la première phase, une route est construite par une suite d'insertions doubles, suivie d'insertions simples. Dans une insertion double, deux sommets candidats i et j sont insérés entre deux sommets consécutifs r et s de la route courante, de manière à maximiser :

$$\frac{S_i + S_j}{c_{ri} + c_{ij} + c_{js} - c_{rs}}$$

Cette méthode se comporte mieux que l'insertion simple et permet de réduire le nombre d'itérations. Lors d'une insertion, on tolère aussi une violation de la contrainte de coût maximal, en autant qu'elle ne dépasse pas λC_{max}, avec $\lambda \geq 1$. Puis, la solution est améliorée (et peut redevenir réalisable) avec des routines de recherche locale 2-opt et 3-opt [46]. La troisième phase tente ensuite de réduire la longueur de la tournée courante par remplacement de sommets. Enfin, la phase finale essaie d'inclure des sommets non encore visités. Les trois dernières étapes sont répétées dans l'espoir de trouver une solution de meilleure qualité.

Une méthode de recherche locale est rapportée dans [17] où un ensemble de tournées précédemment rencontrées est maintenu en mémoire afin d'améliorer la meilleure tournée courante. Deux types de modifications sont proposées. Dans la procédure *two-point exchange*, chaque sommet dans la meilleure tournée courante est considéré à tour de rôle et est échangé avec un sommet dans une des autres tournées conservées en mémoire. Dans la procédure *one-point*, un sommet est

ajouté à la meilleure tournée courante. Il faut noter que l'heuristique n'est pas une méthode de descente pure, car une légère détérioration de l'objectif est autorisée.

1.3.2 Métaheuristiques

Dans [30], les auteurs proposent une recherche tabou pour le STSP. Les mouvements consistent à insérer ou à supprimer des groupes de sommets selon la valeur d'un ratio entre le gain et le détour. La suppression est favorisée par un faible ratio alors que l'insertion est favorisée par un ratio élevé. A chaque itération, un paramètre reflétant l'importance à attribuer au coût et au gain d'une tournée, est modifié de manière à favoriser l'insertion ou la suppression selon que la tournée courante est réalisable ou non. Dans cet article, on rapporte aussi une procédure d'insertion de groupes de sommets. Ici, une mesure de dispersion et de proximité pour un groupe de sommets est calculée en agrégeant le groupe en un seul sommet dont le gain est égal à la somme des gains des sommets qu'il contient.

D'autres types de métaheuristiques ont aussi été rapportées pour l'OP, comme les algorithmes génétiques [72], les systèmes de colonies de fourmis [44] et les réseaux de neurones [76]. Dans l'algorithme génétique décrit dans [72], une population de solutions réalisables est d'abord générée de façon aléatoire. Les auteurs utilisent ensuite un opérateur de croisement où une sous-séquence de sommets provenant de l'un des parents est insérée à l'intérieur de la séquence de l'autre parent (injection). Afin de diversifier la population, un opérateur de mutation fait appel à des opérateurs d'ajout, de remplacement et d'échange de sommets. Il faut noter qu'une fonction objective avec pénalités est utilisée afin de permettre d'explorer le domaine non réalisable.

Dans [76], les auteurs appliquent un réseau de neurones de type Hopfield pour la résolution du problème OP. Chaque ligne dans le réseau de neurones correspond

à un sommet et chaque colonne correspond à une position du sommet dans le chemin. Ainsi, $v(i,j)$ est le niveau d'activation du noeud (i,j) et sa valeur est soit 0, soit 1. Pour tout $j \leq N$, si $v(i,j) = 1$, alors le sommet i est le j^e sommet visité dans la solution. Une dernière colonne est ajoutée pour calculer le score total de la solution. La fonction d'énergie à minimiser est composée de 6 termes permettant de pondérer la qualitè de la solution et sa réalisabilité (i.e., chaque sommet doit être visité une fois exactement). Les poids sur les liens entre les neurones ne sont pas constants et correspondent à la deuxième dérivée partielle de la fonction d'énergie. Le modèle inclut aussi une procédure d'insertion et une routine 2-opt pour améliorer les solutions produites par le réseau de neurones.

1.3.3 Méthodes exactes

Les procédures de résolution exactes proposées pour le STSP s'appuient sur un arbre d'énumération des solutions, géré efficacement par l'évaluation de bornes. Ces bornes sont souvent obtenues à partir de formulations linéaires en nombres entiers du problème.

Les auteurs dans [42] proposent le calcul d'une borne supérieure via la résolution d'un problème de type sac à dos. Des poids W_i sont associés à chaque sommet i, en l'occurrence :

$$W_i = \alpha \min_{j \neq i} c_{ji} + (1 - \alpha) \min_{k \neq i} c_{ik}$$

où $0 \leq \alpha \leq 1$. Avec cette définition, où l'on considère une combinaison linéaire des arcs de coût minimum entrants et sortants de chacun des sommets, le coût d'une tournée est nécessairement supérieur à la somme des poids des sommets visités. Ainsi, la solution optimale du problème de sac à dos binaire appliqué aux sommets $i \in V \setminus \{1\}$ afin de maximiser le gain total, avec un poids limité à $C_{max} - W_1$, est une borne supérieure pour le STSP. Cette borne supérieure est utilisée dans un algorithme de branch-and-bound.

Dans [29] et [23], les auteurs proposent un algorithme de branch-and-cut fondé sur des classes d'inégalités valides. Le premier algorithme permet de résoudre des problèmes comportant jusqu'à 300 sommets, le deuxième jusqu'à 500 sommets.

Une relaxation de type 1-arbre (arbre enraciné au dépôt, plus un arc additionnel de retour au dépôt), déjà proposée pour le problème du voyageur de commerce, est adaptée dans [59] à l'aide d'une relaxation lagrangienne des contraintes liées au coût maximal d'une tournée et au degré des sommets. Cette relaxation est exploitée à l'intérieur d'un algorithme de branch-and-bound.

Righini et al. [61] s'attaquent à un problème d'orientation en appliquant un algorithme de plus court chemin élémentaire avec contraintes de ressources résolu par la programmation dynamique bi-directionnelle. Un chemin est étendu de deux façons, à partir du début et en partant de la fin. Les deux chemins seront fustionnés dès qu'ils se rencontrent au milieu. Les chemins dominés sont écartés et une technique de réduction d'états est utilisée.

1.4 Problèmes multi-véhicules

1.4.1 Heuristiques

Comme c'est le cas dans le contexte du TSP, quelques heuristiques pour le STSP peuvent être étendues au cas multi-véhicules. Par exemple, les auteurs dans [18] proposent une extension directe d'une heuristique développée pour l'OP au problème d'orientation par équipes. Butt et Cavalier [13] s'intéressent pour leur part à un problème de recrutement d'athlètes universitaires, qui peut être modélisé comme un problème avec gains impliquant plusieurs véhicules. Le recruteur dans une université doit recruter des joueurs de football dans les universités

environnantes en un nombre limité de jours. Chaque jour, il dispose d'un temps limite pour visiter quelques universités selectionnées et revenir à son université d'attache. Une mesure de gain est associé à chaque université en fonction de son potentiel de recrutement. L'objectif est de visiter un ensemble d'universités maximisant le potentiel de recrutement sur l'ensemble des journées disponibles. Les auteurs proposent une approche de résolution heuristique fondée sur une mesure W_{ij} qui évalue l'intérêt de visiter deux sommets i et j dans une même tournée :

$$W_{ij}(\alpha) = \alpha \frac{g_i + g_j}{T_{max}} t(1, i, j, 1) + (1 - \alpha) \frac{g_i + g_j}{t(1, i, j, 1)} T_{max},$$

où $0 \leq \alpha \leq 1$, $t(1, i, j, 1)$ est la durée de la tournée visitant uniquement les sommets i et j et T_{max} est la durée maximale d'une tournée. W_{ij} met en valeur soit les sommets de gain élevé et qui sont proches du dépôt, soit les sommets de gain élevé mais qui sont plus distants, dépendemment de la valeur de α. Les tournées sont construites une à une en insérant les sommets appartenant au couple de sommets de valeur maximale.

1.4.2 Métaheuristiques

La première métaheuristique qui a été développée pour la version multi-véhicules est celle rapportée dans [71]. Il s'agit d'une recherche taboue avec mémoire adaptative.

Dans [75], le problème de planification des sites touristiques à visiter a été modélisé comme un problème d'orientation avec plusieurs équipes. Chaque site a un score, un temps de service et une fenêtre de temps indiquant les heures d'ouverture et de fermeture du site. L'approche de résolution est une recherche locale qui se divise en une étape d'insertion suivie d'une étape de perturbation afin d'échapper aux minima locaux.

Dans [69], les auteurs résolvent de problème d'orientation avec plusieurs équipes

mais sans fenêtres de temps. Ils utilisent la méthode GRASP avec Path Relinking. Ils comparent la performance de leur algorithme avec celui décrit dans [49], qui est une métaheuristique basée sur les colonies de fourmis.

Une recherche à voisinage variable combinée avec deux schémas différents de recherche tabou a été proposée dans [2] pour deux variantes de problèmes de tournées avec gains : le problème d'orientation en équipes avec contraintes de capacité (CTOP pour *Capacitated Team Orienteering Problem*) et le problème de tournées profitables avec contraintes de capacité (CPTP pour *Capacitated Profitable Tour Problem*). La recherche à voisinage variable permet de changer la structure de voisinage, lorsque la recherche tabou ne progresse plus avec le voisinage courant.

1.4.3 Méthodes Exactes

Dans [14] et [34], les auteurs adaptent la formulation classique de partitionnement d'ensembles, telle qu'utilisée dans les procédures de résolution du VRP faisant appel à la génération de colonnes, à deux problèmes de tournées avec gains multi-véhicules. Pour Butt et Ryan, la flotte de véhicules est hétérogène (chaque type de véhicule a une durée maximale). Dans le travail de Gueguen, la flotte est homogène de plus, une contrainte de capacité est associée aux véhicules et des fenêtres de temps sont associées aux clients.

Pour la résolution du sous-problème, les auteurs dans [14] commencent par ordonner les sommets selon leur gain réduit. Les sommets avec un gain réduit positif sont mis de côté car ils ne peuvent apporter d'amélioration. Le principe de l'algorithme est de parcourir tous les sous-ensembles de sommets contenant le dépôt et de résoudre le problème du voyageur de commerce correspondant lorsque le gain associé est suffisamment important. Les sous-ensembles sont explorés en

ajoutant progressivement des sommets aux sous-ensembles courants. Ainsi, quand la solution à un problème du voyageur de commerce donné dépasse la durée limite, les sous-ensembles fils ne sont plus explorés. Les problèmes du voyageur de commerce sont résolus à l'aide d'une procédure de résolution exacte déjà existante. Au niveau de l'algorithme de branch-and-price, la résolution à chaque noeud se fait par la méthode de génération de colonnes et le branchement est réalisé sur des couples de sommets. Le couple (i, j) choisi est celui qui maximise :

$$\sum_{k\in\Omega} a_{ik}a_{jk}x_k \qquad \text{avec} \qquad \sum_{k\in\Omega} a_{ik}a_{jk}x_k < 1,$$

où Ω est l'ensemble des colonnes (tournées) courantes et a_{ik} est égal à 1 si le sommet i appartient à la tournée k, et 0 sinon . En cas d'égalité, le choix se fait sur le couple de sommets ayant le ratio suivant le plus élevé :

$$\frac{g_i + g_j}{\min\{c_{oi} + c_{ij} + c_{j0}, c_{0j} + c_{ji} + c_{i0}\}}$$

Les auteurs mentionnent que le branchement sur un couple de variables permet de réduire le nombre de noeuds dans l'arbre de branchement.

Dans [34], le sous-problème est formulé comme un problème de plus court chemin élémentaire avec contraintes de ressources. Ce dernier est résolu par programmation dynamique en faisant appel à un critère de dominance pour réduire le nombre d'états. Un algorithme de branch-and-price avec branchement sur les arcs est utilisé afin d'obtenir une solution entière au problème (i.e., un arc avec une valeur fractionnaire est soit imposé, soit interdit).

Feillet [9] formule aussi le sous-problème comme un problème de plus court chemin élémentaire avec contraintes de ressources. L'algorithme de branchement s'intéresse d'abord aux variables implicites $y_i = \sum_{r\in\Omega} a_{ir}x_r$. Si un sommet i est visité un nombre fractionnaire de fois, deux branches sont dérivées pour le cas où le client est servi et le cas où le client n'est pas servi. Ensuite, l'algorithme branche sur les carcs. Deux cas sont alors à considérer :

– si aucun des deux sommets i et j n'est contraint à être visité, alors trois branches sont dérivées. La première interdit le sommet i. La seconde branche force la visite de i et l'utilisation de l'arc (i, j). La troisième branche force la visite de i et interdit l'utilisation de l'arc (i, j).
– si i ou j sont contraints à être visités, alors deux branches sont dérivées, soit une branche où la visite de l'arc (i, j) est forcée, et l'autre branche où la visite est interdite.

1.5 Problèmes de tournées avec réutilisation de véhicules

Le problème de tournées avec réutilisation de véhicules a été peu étudié dans la littérature. On ne peut citer que quelques travaux comme [10, 11, 24, 51, 52, 70, 79].

Dans [70], les auteurs proposent une heuristique pour la résolution d'un tel problème sans fenêtres de temps. Cette heuristique se divise en trois phases. Dans la phase 1, un ensemble de routes pour le problème classique (sans réutilisation de véhicules) est généré en utilisant un algorithme de recherche tabou [62]. Dans la phase 2, on sélectionne un sous-ensemble de routes générées en phase 1, en favorisant les routes appartenant à de bonnes solutions. A partir de ce sous-ensemble, on génère l'ensemble K de toutes les solutions possibles au problème classique de tournées de véhicules avec une approche énumérative. Dans la phase 3, on tente d'obtenir une solution réalisable au problème avec réutilisation de véhicules, en résolvant un problème de mise en boîte (ou *bin-packing*) pour chaque solution k dans l'ensemble K, où $f_{k,l}$ est la durée de la l^e route pour $l = 1, \ldots, m_k$, et m_k est le nombre de routes dans la solution k. Une solution consite à placer les m_k articles (routes) de poids $f_{k,1}, \ldots, f_{k,m_k}$ dans m boîtes de taille M, où m est

le nombre de véhicules disponibles ($m << m_k$) et M est la durée maximale d'une journée de travail pour un véhicule. Pour identifier de telles solutions, les articles sont considérés en ordre décroissant de leur poids et sont affectés à tour de rôle à la boîte dont le poids accumulé est minimal, ce qui est similaire à l'heuristique *Worst-Fit*. Si aucune des m boîtes n'a un poids accumulé qui dépasse M une fois que tous les articles sont placés, alors on a une solution. Sinon, on échange des articles appartenant à différentes boîtes dans l'espoir de trouver une solution. Cependant, il n'est pas garanti qu'une solution réalisable soit trouvée de cette façon. On peut donc étendre les solutions admissibles au domaine non réalisable, en permettant d'excéder la durée maximale d'une journée de travail, moyennant une pénalité dans l'objectif.

Dans [10], les auteurs décrivent une heuristique de recherche tabou pour le problème avec réutilisation de véhicules et fenêtres de temps, et d'autres contraintes telles que la présence de plusieurs types de véhicules avec contraintes d'accès à certains clients. De plus, les chauffeurs doivent respecter le temps légal de conduite par jour et peuvent avoir soit trois pauses d'une demi-heure ou deux pauses de 45 minutes. Les temps de déchargement sont également pris en compte. L'algorithme se divise en trois phases. La solution initiale de chaque phase est le résultat de la phase précédente. En phase 1, on tente de construire une solution qui ne respecte pas nécessairement les fenêtres de temps tout en minimisant les temps de parcours. En phase 2, deux objectifs sont considérés en même temps : le respect des fenêtres de temps et le temps maximal de conduite. L'objectif de la phase 3 est de réduire le coût de la solution tout en maintenant la réalisabilité. Deux types de mouvements sont appliqués : l'insertion d'un nouveau sommet et l'échange de deux sommets.

L'algorithme défini dans [50] est similaire à celui proposé dans [70]. Une mémoire adaptative est utilisée afin de sauvegarder les meilleures solutions rencontrées. A chaque itération, une nouvelle solution est construite à partir des

routes qu'on retrouve dans la mémoire, puis cette solution est améliorée à l'aide d'une recherche tabou. La solution obtenue à la fin peut être stockée dans la mémoire si elle est suffisamment bonne. La mémoire est organisée de façon à ce que les meilleures solutions apparaissent en premier. Ainsi, puisque des solutions non réalisables peuvent être visités pendant la recherche, la position d'une solution dépend d'abord d'une mesure de non réalisabilité. Pour deux routes ayant une même mesure de non réalisabilité, celle de plus petit coût est considérée en premier. Le problème d'affectation est résolu par une heuristique de mise-en-boîte. Il faut noter qu'il n'est aucunement garanti qu'une solution réalisable soit obtenue à la fin de l'algorithme.

Récemment, un algorithme de recherche adaptative guidée a été proposé pour le problème avec fenêtres de temps et réutilisation de véhicules avec contraintes d'incompatibilé véhicule-client, dans le cas d'une application dans la distribution de produits [7]. Une approche par décomposition génère des sous-problèmes plus simples qui sont résolus par des heuristiques spécifiques. Le mécanisme d'adaptation est basé sur la pénalisation d'intervalles de temps dits critiques, soit des intervalles de temps où plusieurs routes sont actives.

Bien que les travaux sur le problème avec réutilisation de véhicules sont rares, on retrouve de nombreuses applications dans le monde réel, en particulier avec l'avènement du commerce électronique qui offre des services de livraison à domicile. L'essor du commerce électronique s'est accompagné d'une intense activité de recherche, se traduisant par de nombreuses études à la fois théoriques et empiriques. Par exemple, ECOMLOG est un regroupement de chercheurs de l'Université de Helsinki qui a fait une étude détaillée des services offerts par les détaillants en ligne. Ils ont produit, entre autres, plusieurs articles sur les épiceries en ligne. Dans [37, 57, 67, 78] les auteurs ont étudié différentes façons de recevoir, traiter et livrer les commandes, et leurs effets sur les coûts. Dans ces études, les services offerts sont caractérisés par :

1. les coûts de livraison,
2. les heures et les délais de livraison,
3. les durées des fenêtres de temps
4. le modèle de distribution.

Chaque détaillant en ligne a sa propre politique de gestion, que ce soit pour les délais admissibles ou les coûts de livraison. Dans la plupart des cas, le client doit finaliser sa commande au moins un jour avant la livraison. Par exemple, Peapod.com demande à ses clients de finaliser leurs commandes avant minuit s'ils veulent être livrés le lendemain. Dans ces études, Webvan.com était le seul détaillant en ligne qui offrait des services de livraison le même jour. La plupart des détaillants ont leur propre flotte de véhicules, mais on retrouve quelques exceptions comme NetGrocer.com, qui utilise les services de Federal Express pour livrer ses commandes.

Les commandes peuvent être livrées à partir d'un centre de distribution créé spécialement pour le service ou à partir d'un centre d'achats existant. Dans ce dernier cas, la livraison à domicile est moins coûteuse et requiert un capital initial moins important. Les auteurs dans [78] prétendent toutefois que le modèle du centre d'achats n'est pas efficace opérationnellement. Ils suggèrent de commencer par ce modèle pour éviter un grand investissement au départ et d'adopter par la suite le modèle du centre de distribution dédié.

Un autre aspect important est la durée des fenêtres de temps. Par exemple, chez Webvan.com, les clients choisissent une fenêtre de temps de 30 minutes et il y a une pénalité de retard pour le livreur et une autre pénalité pour le client s'il n'est pas présent au moment de la livraison. Évidemment, plus la fenêtre de temps est serrée, plus les coûts de livraison sont élevés. Dans [67], les auteurs ont fait des simulations avec deux modèles spécifiques : Streamline.com qui adopte le service non attendu et Webvan.com qui adopte le service attendu avec un délai de livrai-

son d'une demi-heure. Ils montrent que les coûts de livraison de Webvan.com sont cinq fois plus élevés. Le service non attendu diminue considérablement les coûts en permettant de choisir la fenêtre de temps la plus profitable. Punakivi [55] étudie l'utilisation de routes fixes versus l'utilisation de routes optimisées. La simulation a révélé que les diminutions de coûts avec les routes optimisées sont de l'ordre de 18% pour les régions denses et vont jusqu'à 54% pour les régions de moindre densité. Punakivi [55] propose aussi d'effectuer des tournées de durée limitée et d'utiliser au maximum la capacité des véhicules. Il montre que plus le client a le contrôle sur la sélection de sa fenêtre de temps, plus les coûts d'opérations sont élevés.

Un autre aspect important est la tarification des livraisons. Initialement, les livraisons étaient gratuites peu importe la quantité commandée. Par la suite, des frais de livraison ont été imposés aux commandes dont la valeur ne dépassait pas un certain montant. De nos jours, les frais de livraison varient souvent selon la plage à laquelle appartient la valeur de la marchandise commandée. Par exemple, le coût de livraison chez Webvan.com est de 9.95 $ pour les commandes de moins de 75 $ et il n'y a pas de frais pour les commandes de plus de 100 $. Un autre exemple est Peapod.com. Sa structure de tarification est similaire à celle de Webvan, mais le montant minimum requis pour une livraison est de 50 $.

Les services de livraison à domicile peuvent se modéliser comme des problèmes de tournées avec gains, puisque la flotte de véhicules est souvent limitée et qu'on peut associer un gain ou profit à la commande de chaque client.

1.6 Problèmes Dynamiques

Le routage et la répartition dynamique de véhicules représente une vaste classe de problèmes où des informations sont révélées en temps réel au planificateur.

Ces problèmes ont été grandement étudiés en recherche opérationnelle au cours des dernières années. Dans cette section, nous revoyons brièvement les principaus travaux réalisés en routage et répartition dynamique de véhicules ainsi que les méthodologies générales adoptées pour traiter de l'aspect dynamique de ces problèmes. Dans un contexte dynamique, certains éléments du problème sont modifiés ou révélés durant l'exécution des routes, ce qui nécessite une replanification en réaction à ces changements. Dans un contexte stochastique, l'occurence des événements peut être caractérisée par une distribution de probabilités connue.

Dans la littérature, il existe différentes façons de planifier dans un contexte dynamique. Certaines approches sont purement réactives, et appliquent des heuristiques d'insertion et une ré-optimisation à chaque occurrence d'un nouvel évenement. D'autres exploitent des informations probabilistes sur les événements futurs afin d'obtenir des plans plus robustes.

1.6.1 Approches sans anticipation

Les premiers travaux dans le domaine n'utilisent aucune information sur le futur ou le font de façon très limitée. Dans [43], les auteurs ont étudié différentes politiques de repositionnement des véhicules en vue de mieux servir les clients à venir. Ces politiques ne font toutefois aucunement appel à une connaissance quelconque sur l'arrivée des clients futurs.

Dans [47], les auteurs décrivent une approche où deux objectifs différents sont utilisés. Un premier objectif à court terme est appliqué à la première portion de la route planifiée. Le second objectif à long terme est appliqué à la dernière portion de la route planifiée (i.e. aux clients servis plus loin dans le temps). Ce second objectif a pour but de créer des blocs de temps d'attente dans la route, de façon à faciliter l'insertion des clients futurs. La stratégie avec un horizon

double est meilleure que celle avec un seul horizon selon des tests réalisés avec des compagnies de livraison de courrier à Vancouver.

Dans [28], les auteurs utilisent une approche hybride pour résoudre un problème dynamique de tournées avec fenêtres de temps. Tout nouveau client est rapidement inséré dans la solution courante, puis une recherche tabou avec mémoire adaptative est appliqué afin d'améliorer cette solution jusqu'à l'arrivée du prochain événement. La structure de voisinage est basée sur l'opérateur de CROSS exchange où deux segments de routes sont échangés. Une implantation parallèle de la recherche tabou est aussi réalisée afin d'accélérer les temps de calcul et permettre une réoptimisation plus poussée entre deux événements. Le parallélisme se situe à deux niveaux. Premièrement, des fils (ou *threads*) de recherche sont exécutés en parallèle, à partir de solutions de départ différentes, produites à l'aide de la mémoire adaptative. Deuxièmement, à l'intérieur de chaque fils de recherche, des recherches tabous sont appliquées en parallèle à chacun des sous-problèmes générés par une méthode de décomposition qui partitionne l'ensemble des routes dans la solution courante en plusieurs sous-ensembles de routes (sous-problèmes). Cette approche permet de réaliser une intensification de la recherche, car chaque recherche tabou s'attaque à un sous-problème de plus petite taille. Étant donné la plate forme disponible, en l'occurence un réseau de stations de travail, un schéma maître-esclave a été retenu. Le maître gère la mémoire adaptative et crée les solutions de départ pour les processus tabous esclaves. Dans ce travail, deux types d'événements interrompent la recherche tabou :

1. L'arrivée d'une nouvelle requête.
2. La fin du service à un client.

Quand une nouvelle requête est reçue, les processus tabous sont arrêtés et retournent la meilleure solution trouvée au processus maître pour inclusion dans la mémoire adaptative (si la solution est suffisamment bonne). La nouvelle requête est ensuite insérée dans chacune des solutions en mémoire adaptative. Par la

suite, de nouvelles solutions de départ sont générées à l'aide de la mémoire adaptative afin de réinitialiser les processus tabous. D'autre part, lorsqu'un véhicule a teminé son service au client courant, les processus de recherche tabou sont également interrompus. On indique alors au véhicule le prochain client à visiter selon la meilleure solution conservée dans la mémoire adaptative. Après avoir mis à jour les autres solutions en mémoire pour qu'elles reflètent la situation courante, de nouvelles solutions de départ sont générées avec la mémoire adaptative afin de réinitialiser les recherches tabous. Dans [27], les auteurs ont étendu cette approche à un problème dynamique avec cueillettes et livraisons en faisant appel à une structure de voisinage basée sur des chaînes d'éjection.

Dans [77], les auteurs traitent aussi d'un problème dynamique avec cueillettes et livraisons où la gestion des requêtes dynamiques est réalisée sans connaissance sur le futur. Les auteurs comparent une heuristique simple d'insertion avec deux approches de réoptimisation plus sophistiquées. Ils concluent que :

- Une réoptimisation plus poussée améliore beaucoup les résultats par rapport à une simple heuristique d'insertion.
- Lorsque le degré de dynamique augmente, il est crucial de réoptimiser.

1.6.2 Approches avec anticipation

Cette section présente des travaux qui exploitent une connaissance probabiliste sur les événements futurs afin de produire des plans de routage plus robustes.

Dans [8], les auteurs considèrent une version dynamique du problème de tournées de véhicules avec fenêtres de temps où les nouvelles requêtes se présentent dans le temps et l'espace selon une certaine loi de probabilité. L'idée de base est de maintenir un ensemble de plans de routage consistants avec les décisions prises depuis le début, tout en suivant le plan courant, soit celui qui a le plus de simila-

rités avec les autres, et ce, jusqu'à l'occurence du prochain événement. Des actions sont déclenchées lorsqu'un de ces événements survient :

1. *Génération de plans* : survient lorsque la recherche locale a terminé l'optimisation d'un certain plan. Cet événement peut entraîner la sélection d'un nouveau plan courant.
2. *Arrivée d'une nouvelle requête* : cette requête peut être rejetée si aucun plan ne peut l'accommoder ; autrement, la nouvelle requête est insérée à moindre coût dans tous les plans qui permettent l'insertion, et ces plans forment alors un nouvel ensemble.
3. *Départ d'un véhicule* : survient lorsque le plan courant requiert le départ du véhicule de sa position courante. Tous les plans incompatibles avec cette décision doivent alors être supprimés.

Une seconde idée clé est l'anticipation des requêtes futures en procédant à un échantillonnage des variables aléatoires qui représentent ces requêtes. Les solutions sont ainsi optimisées en tenant compte de plusieurs réalisations possibles des requêtes futures (ce que les auteurs appellent des scénarios).

Dans [35], une approche à base de scenarios est également utilisée. Une journée est divisée en intervalles, et les changements à la solution courante ne sont apportés qu'à la fin d'un intervalle. Un échantillonage est effectué à chaque intervalle pour générer de nouveaux scénarios. Un algorithme de branch-and-bound est utilisé afin d'évaluer si une requête doit être affecté à la solution dans l'intervalle courant ou si on doit reporter la décision au prochain intervalle. Ceci est fait en résolvant le problème à l'aide de scénarios générés avec les deux alternatives.

Dans [36], les auteurs exploitent une distribution de probabilités connue sur les requêtes futures en forçant un véhicule à attendre au client courant si la probabilité qu'une nouvelle requête arrive bientôt à proximité du véhicule excède un certain seuil.

Il faut enfin noter un travail réalisé par Campbell et Savelsbergh [16] pour un contexte dynamique particulier. Les auteurs s'intéressent ici au problème de livraison à domicile pour lequel la décision d'accepter ou non une nouvelle requête doit être prise sur le champ (bien que les routes elles-mêmes ne seront exécutées que beaucoup plus tard).

Dans ce travail, on suppose que les clients potentiels sont connus à l'avance et que chaque client a une probabilité p_i d'appeler. Cette probabilité décroît linéairement avec le temps jusqu'à la fin de l'horizon de planification. A chaque commande d'un client est associée une demande d_i et un gain ou revenu r_i. L'objectif est de maximiser le profit, soit le revenu total moins les coûts de livraison. Les auteurs résolvent un problème de tournées de véhicules à chaque fois qu'une nouvelle requête se matérialise, en incluant les requêtes déjà acceptées, la nouvelle requête et les requêtes anticipées. Si le profit espéré sans cette requête est plus grand que le profit espéré en incluant la requête, alors cette dernière est rejetée, sinon elle est acceptée dans l'intervalle de temps associé au plus grand profit espéré. Pour les requêtes futures (i.e., non encore matérialisées), les demandes et les revenus sont ajustés en fonction de leurs probabilités.

Les auteurs proposent différents critères d'insertion pour estimer le coût espéré d'une route sans la requête et avec la requête avant de décider de l'acceptation ou du refus de cette dernière. L'estimation du coût d'insertion d'une requête future j entre les requêtes $i-1$ et i se fait selon les critères suivants (où r_j est le revenu associé à la requête j et c_{ij} est le coût de l'arc (i,j)) :

1. profit simple : $r_j - (c_{i-1,j} + c_{ji} - c_{i-1,i})$,
2. profit actualisé : $p_j r_j - (c_{i-1,j} + c_{ji} - c_{i-1,i})$

Dans les deux critères précédents, les requêtes $i-1$ et i peuvent être des requêtes véritables ou non. Le profit actualisé est donc une façon simple de distinguer le profit selon qu'il s'agit d'une requête réelle (pour laquelle la probabilité associée est de 1) ou une requête anticipée. D'autres critères plus sophistiqués ont

également été proposés à cet effet, plus précisément :

3. Soient u et v deux requêtes déjà acceptées. La contribution d'une arête (j,k) au coût espéré du segment de route compris entre u et v est :

$$c_{j,k}p_jp_k \prod_{l=j+1,\ldots,k-1}(1-p_l)$$

La contribution du segment de routes (u,v) est donné par :

$$\sum_{j=u,\ldots,v-1}\sum_{k=u+1,\ldots,v} c_{j,k}p_jp_k \prod_{l=j+1,\ldots,k-1}(1-p_l)$$

où c_{jk} est le coût de l'arête (j,k), p_j et p_k sont les probabilités de réalisation des requêtes j et k, et $(1-p_l)$ est la probabilité de non occurrence de la requête l.

Le coût pour l'insertion de la requête l entre u et v est calculé comme étant la différence entre le coût espéré du segment de route (u,v) avec et sans la requête l.

4. Ici, on estime d'abord le coût d'insertion v_1 d'une requête j entre deux sommets voisins quelconques $i-1$ et i, puis le coût v_2 entre deux sommets voisins déjà acceptés $k-1$ et k :

$$v_1 = c_{i-1,j} + c_{j,i} - c_{i-1,i}$$

$$v_2 = c_{k-1,j} + c_{j,k} - c_{k-1,k}$$

Etant donné que le coût de base est v_2, et que la différence entre v_2 et v_1, pondérée par les probabilités de présence des requêtes $i-1$ et i, est $p_{i-1}p_i(v_2-v_1)$, le profit espéré en insérant j est estimé par :

$$p_jr_j - (v_2 - p_{i-1}p_i(v_2-v_1)).$$

1.7 Plan de la thèse

Les différentes variantes des problèmes de tournées avec gains que nous avons évoquées dans ce chapitre prennent rarement en compte de véritables contraintes de capacité (en termes de volume et de poids, par exemple) et ne considèrent pas toujours les fenêtres de temps aux clients. Ces éléments permettent pourtant de rendre ces problèmes encore plus réalistes. Nous proposons donc dans cette thèse d'étendre le problème du voyageur de commerce avec gains au problème multi-véhicules avec fenêtres de temps où chaque véhicule peut être réutilisé afin d'exécuter plusieurs tournées au cours de sa journée d'opérations. L'objectif étant de déterminer un ensemble de journées de travail pour les véhicules qui maximise le profit total, soit le gain total associé à l'ensemble des clients desservis moins les coûts de transport.

Cette thèse se compose d'une introduction, d'une conclusion et de trois articles. Le chapitre 2 présente d'abord un algorithme exact que nous avons développé pour résoudre le problème de tournées de véhicules avec fenêtres de temps et réutilisation des véhicules. À notre connaissance, il s'agit de la première approche exacte conçue pour ce problème. L'algorithme est basé sur une méthode de génération de colonnes où le sous-problème correspond à un plus court chemin élémentaire avec contraintes de ressources. Afin d'accélérer l'algorithme, chaque sous-problème est d'abord résolu de façon heuristique en relaxant la règle de dominance entre les chemins générés. La résolution exacte du sous-problème prend effet seulement lorsque l'heuristique n'est plus en mesure d'identifier de chemins avantageux. De plus, une méthode de stabilisation, déjà rapportée dans la littérature, a été utilisée afin d'améliorer les performances de l'algorithme. Des résultats expérimentaux sont rapportés pour des instances allant jusqu'à 40 clients. L'article qui correspond à ce chapitre a été publié en 2010, dans le volume 202 de la revue *European Journal of Operational Research*.

Le chapitre 3 traite d'une méthode heuristique pour la résolution du même problème. Cette heuristique est une recherche adaptative à grand voisinage où plusieurs opérateurs de retrait et d'insertion de clients ont été développés et adaptés au problème qui présente une structure hiérarchique à trois niveaux, soit les journées de travail, les routes et les clients. On retrouve donc des opérateurs différents selon les niveaux, ce qui constitue une innovation par rapport à l'approche classique qui n'opère qu'au niveau client. Un mécanisme adaptatif ajuste les poids des opérateurs agissant aux différents niveaux afin de choisir les opérateurs à appliquer à chaque itération, au sein d'une structure globale de recherche qui alterne entre les différents niveaux de manière cyclique. De plus, lors de l'insertion d'un client, une route dans la journée de travail d'un véhicule peut être divisée en deux sous-routes, s'il n'y a aucune autre alternative réalisable, afin d'intégrer le plus de clients possibles dans la solution. Des résultats numériques avec des instances allant jusqu'à 1000 clients sont rapportés. L'article qui correspond à ce chapitre a été soumis pour publication à la revue *Computers & Operations Research*.

Le chapitre 4 est consacré à la résolution d'une version dynamique du problème où les requêtes des clients arrivent de façon continue au cours d'une journée d'opérations. Une stratégie d'acceptation ou de refus de chaque nouvelle requête est développée en tenant compte de sa profitabilité espérée. Pour ce faire, un ensemble de scénarios possibles quant à l'occurrence des requêtes futures est généré et une nouvelle requête est acceptée lorsqu'elle apparaît profitable sur l'ensemble des scénarios. Des résultats numériques démontrent les bénéfices de cette approche par rapport à une approche myope où les requêtes futures ne sont aucunement tenus en compte. L'article qui correspond à ce chapitre a été soumis pour publication dans *Annals of Operations Research*

En terminant, il faut observer que dans les chapitres 2 et 3, le gain est fixé à 1 pour chacun des clients et est ensuite multiplié par une constante arbitrairement

grande, de telle sorte que l'objectif hiérarchique classique suivant est obtenu : d'abord, maximiser le nombre de clients servis ; ensuite, pour le même nombre de clients, minimiser la distance totale parcourue. La notion de gain est toutefois pleinement exploitée au chapitre 4, alors que les gains diffèrent selon les clients et interviennent au niveau de la stratégie d'acceptation ou de refus des nouvelles requêtes.

Chapitre 2

An Exact Algorithm for a Vehicle Routing Problem with Time Windows and Multiple Use of Vehicles

Résumé. Le problème de tournées avec utilisation multiple des véhicules apparaît lorsque chaque véhicule réalise plusieurs tournées durant sa journée dû à des limitations sur la durée des tournées (par exemple, lorsque des denrées périssables sont transportées). Les tournées visitent des clients qui se caractérisent par un gain, une demande et une fenêtre de temps. Nous supposons une flotte de véhicules de taille fixe, ce qui signifie que la demande peut excéder la capacité de la flotte. Les clients sont alors choisis sur la base du gain associé moins les coûts de transport. Une approche de type branch-and-price est proposée pour résoudre ce problème où les bornes inférieures sont obtenues en résolvant la relaxation linéaire du problème. Les sous-problèmes correspondent à des problèmes de chemins plus courts élémentaires avec contraintes de ressources. Des résultats

numériques sont présentés pour des problèmes euclidiens dérivés d'instances classiques du problème de tournées de véhicules avec fenêtres de temps. Des résultats additionnels sont rapportés pour des instances plus contraintes où les fenêtres de temps sont réduites.

An Exact Algorithm for a Vehicle Routing Problem with Time Windows and Multiple Use of Vehicles

Nabila Azi
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Michel Gendreau
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Jean-Yves Potvin
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Cet article a été publié dans *European Journal Of Operational Research*, volume 202, numéro 3, pages 756-763, 2010.

Abstract. The vehicle routing problem with multiple use of vehicles is a variant of the classical vehicle routing problem. It arises when each vehicle performs several routes during the workday due to strict time limits on route duration (e.g., when perishable goods are transported). The routes are defined over customers with a gain, a demand and a time window. Given a fixed-size fleet of vehicles, it might not be possible to serve all customers. Thus, the customers must be chosen based on their associated gain minus the traveling cost to reach them. We introduce a branch-and-price approach to address this problem where lower bounds are computed by solving the linear programming relaxation. The pricing subproblems are elementary shortest path problems with resource constraints. Computational results are reported on Euclidean problems derived from well-known benchmark instances for the vehicle routing problem with time windows. Additional results are reported on more constrained instances where the width of the time windows is reduced.

Keywords : Vehicle routing, time windows, multiple use of vehicles, elementary shortest paths with resource constraints, column generation, branch-and-price.

2.1 Introduction

We consider a variant of the Vehicle Routing Problem with Time Windows (VRPTW) where each vehicle can perform several routes during its workday. Surprisingly, this problem has received little attention in the literature in spite of its importance in practice. For example, in the home delivery of perishable goods, routes are of short duration and must be combined to form a complete workday. We believe that this type of problem will become increasingly important in the future with the advent of electronic services, like e-groceries, where customers can order goods through the Internet. Logistics and socio-economic considerations about different types of home delivery problems, with a particular emphasis on electronic groceries, can be found in [16, 40, 45, 56, 57, 78].

The vehicle routing problem with multiple use of vehicles, but no time windows, has been addressed through heuristic means in [11, 24, 50, 51, 52, 70]. In [70], different solutions to the classical vehicle routing problem are generated using a tabu search heuristic. The routes obtained are then combined to produce workdays for the vehicles by solving a bin packing problem, an idea previously introduced in [24]. Another work by [15] describes insertion heuristics that can efficiently handle different types of constraints including time windows. This is used, in particular, to solve problems with multiple use of vehicles. [51] propose a three-phase heuristic to solve a variant of the problem where, for a given number of vehicles, the objective is to minimize the maximal overtime. Their population-based approach first generates routes with a savings-based heuristic. These routes are then combined to form complete solutions which are finally improved with a local search heuristic. [10] also propose a three-phase heuristic where an initial solution is first produced with an insertion heuristic. This solution is then improved using a tabu search with a neighborhood based on reinsertion and exchange of customers and where infeasibility is allowed. The third phase is aimed at resto-

ring feasibility. In a follow-up work in [11], a more complex variant of the problem with a mixed fleet and maximum overtime constraints is considered. [50] present an adaptive memory-based heuristic, where the memory is made of multiple route solutions. In [3], an exact algorithm is proposed to solve a single vehicle variant of the problem addressed here. This algorithm is a two-phase method in which all non-dominated feasible routes are first generated. Some of these routes are then selected and sequenced to form the vehicle's workday.

In this paper, we extend the work by [3] to the much more challenging multiple vehicle case. To the best of our knowledge, this is the first time that an exact algorithm is proposed for this problem. In Section 4.2, the mathematical programming formulation is presented. The column generation approach is then described in Section 2.3. This is followed in Section 2.4 by a description of the branch-and-price algorithm, where column generation is exploited to solve the subproblem associated with each node in the search tree. Computational results on problem instances derived from Solomon's VRPTW benchmark instances [68] are reported in Section 4.5. Finally, concluding remarks follow in Section 4.6.

2.2 Problem Formulation

Our problem can be stated as follows. We have a set K of vehicles, each of capacity Q, for delivering perishable goods from a depot node to a set of customer nodes $V = \{1, 2, ..., n\}$ in a complete directed graph G. In this graph, a distance d_{ij} and travel time t_{ij} are associated with every arc $(i, j) \in A$, where A is the arc set. Each customer $i \in V$ is characterized by a gain g_i, a demand q_i, a service or dwell time s_i and a time window $[a_i, b_i]$, where a_i is the earliest time to begin service and b_i is the latest time. Thus, a vehicle must wait if it arrives at customer i before a_i. The working day of each vehicle is made of a sequence of routes where

each route starts and ends at the depot (some of these routes might be empty). These routes are denoted by set R, where $|R|$ is large enough to accommodate the maximal number of routes that the fleet can possibly perform in a day. We assume, without loss of generality, that the routes served by any vehicle are numbered in increasing order, that is, a vehicle serves route s after route r if and only if $r < s$.

The depot is denoted by 0 or $n+1$ depending if it is the initial or terminal node of an arc, with $s_0 = s_{n+1} = 0$, $q_0 = q_{n+1} = 0$, $a_0 = a_{n+1} = 0$; $b_0 = b_{n+1} = \infty$; the notation V^+ is used for $V \bigcup \{0, n+1\}$ and A^+ for $A \bigcup \{(0, n+1)\}$, where $(0, n+1)$ is a dummy arc with distance $d_{0,n+1} = 0$ and travel time $t_{0,n+1} = 0$. The duration of each route is limited by forcing the last customer to be served within t_{max} time units of the route start time. Also, a setup time σ^r for loading the vehicle is associated with each route $r \in R$. Here, the setup time is proportional to the sum of service times at each customer along the route.

The following variables are used in the formulation of the problem.

- For each route $r \in R$ and each arc $(i, j) \in A^+$, binary variable x_{ij}^r indicates whether or not arc (i, j) appears in route r (note that when $x_{0,(n+1)}^r = 1$, route r is empty).
- For each route $r \in R$ and each customer $i \in V$, binary variable y_i^r indicates whether or not customer i is served by route r.
- For each route $r \in R$ and each customer $i \in V$, continuous variable t_i^r indicates when the service starts at customer i if it is served by route r. When customer i is not served by route r, this value is meaningless. For each route $r \in R$, t_0^r (resp. t_{n+1}^r) is the time at which the route starts (resp. ends) at the depot.
- For each pair of routes $r, s \in R$ with $r < s$, binary variable z_{rs} indicates whether or not route s immediately follows route r in the workday of one of the vehicles.

The problem is then formulated as follows, with M an arbitrary large constant :

$$\text{Min} \sum_{r \in R} \sum_{(i,j) \in A^+} d_{ij} x_{ij}^r - \alpha \sum_{r \in R} \sum_{i \in V} g_i y_i^r \tag{2.1}$$

s.t.

$$\sum_{j \in V^+} x_{ij}^r = y_i^r, \quad i \in V,\ r \in R, \tag{2.2}$$

$$\sum_{r \in R} y_i^r \leq 1, \quad i \in V, \tag{2.3}$$

$$\sum_{i \in V^+} x_{ih}^r - \sum_{j \in V^+} x_{hj}^r = 0, \quad h \in V,\ r \in R, \tag{2.4}$$

$$\sum_{i \in V^+} x_{0i}^r = 1, \quad r \in R, \tag{2.5}$$

$$\sum_{i \in V^+} x_{i(n+1)}^r = 1, \quad r \in R, \tag{2.6}$$

$$\sum_{i \in V} q_i y_i^r \leq Q, \quad r \in R, \tag{2.7}$$

$$t_i^r + s_i + t_{ij} - M\left(1 - x_{ij}^r\right) \leq t_j^r, \quad (i,j) \in A^+,\ r \in R, \tag{2.8}$$

$$a_i y_i^r \leq t_i^r \leq b_i y_i^r, \quad i \in V,\ r \in R, \tag{2.9}$$

$$t_0^r \geq \sigma^r, r \in R, \tag{2.10}$$

$$t_i^r \leq t_0^r + t_{max}, \quad i \in V,\ r \in R, \tag{2.11}$$

$$\sigma^r = \beta \sum_{i \in V} s_i y_i^r, \quad r \in R, \tag{2.12}$$

$$t_0^s + M(1 - z_{rs}) \geq t_{n+1}^r + \sigma^s, \quad r, s \in R,\ r < s, \tag{2.13}$$

$$\sum_{r \in R} \sum_{s \in R | r < s} z_{rs} \geq |R| - |K| \quad r, s \in R, \tag{2.14}$$

$$x_{ij}^r \in \{0,1\}, \quad i,j \in A,\ r \in R, \tag{2.15}$$

$$y_i^r \in \{0,1\} \quad i \in V,\ r \in R, \tag{2.16}$$

$$z_{rs} \in \{0,1\} \quad r, s \in R,\ r < s, \tag{2.17}$$

$$t_i^r \geq 0, \quad i \in V,\ r \in R. \tag{2.18}$$

The general objective used in this formulation is based on a weighted sum of two conflicting objectives, namely, maximization of total gain versus minimization of total distance. It is obtained by minimizing the difference between the total distance and the total gain, weighted by parameter α (or, conversely, by maximizing the difference between the total gain, weighted by parameter α, and the total distance). This formulation leads to similarities with traveling salesman problems with profits, where the decision to serve a customer request takes into account its associated gain [21]. In the orienteering problem [17, 74], for example, a tour which maximizes the total gain must be found without exceeding some preset threshold on the total distance traveled.

One should note that it might be difficult in practice to find a weight that leads to an appropriate trade-off between the two objectives. No such trade-off analysis is required here, because we consider the usual hierarchical objective, where the number of served customers is first maximized and, for the same number of customers, the total distance is minimized. This objective is easily obtained by setting the gain g_i of each customer $i \in V$ to 1 and by setting α to a sufficiently large value, derived from problem data [3].

In the rest of the formulation, constraints (2.2) and (2.3) state that every customer should be visited at most once. Constraints (2.4), (2.5), and (2.6) are flow conservation constraints that describe the individual routes. Constraint (2.7) states that the total demand on a route cannot exceed vehicle capacity. Constraints (2.8), (2.9), (2.10) and (2.11) ensure feasibility of the time schedule. Note that constraint (2.11) is a deadline constraint for the service at each customer and that constraint (2.9) forces the t_i^r variables to 0 when customer i is not in route r. Consequently, constraint (2.11) is automatically satisfied in this case. Constraint (2.12) defines the vehicle loading time for a route as the sum of the service times over all customers in that route multiplied by a constant β. Constraints (2.13) and (2.14) ensure the proper route sequencing within the

workdays of individual vehicles. In particular, at most $|K|$ workdays can be part of a solution through constraint (2.14), because the number of z_{rs} variables set to one is always equal to the number of routes minus the number of workdays (note that the number of transitions from one route to another in a workday is equal to the number of routes in that workday minus 1).

2.3 Column generation

In practice, the formulation in Section 4.2 is unlikely to be tractable for any instance of reasonable size. We thus address this problem with a column generation approach embedded within a branch-and-price algorithm. Column generation is well documented in the literature. We thus only briefly introduce the definition of the master problem and pricing subproblems in the following. The interested reader will find more details about branch-and-price and column generation in [6, 19, 20].

2.3.1 Master problem

This section introduces a set packing formulation for the master problem (MP), where every column corresponds to a workday. The decision variables x_w are binary variables with value 1 when workday w is in the solution, 0 otherwise.

$$(MP) \qquad \text{Min} \quad \sum_{w\in\Omega}(d_w - \alpha g_w)x_w \tag{2.19}$$

$$\text{s.t.} \quad \sum_{w\in\Omega} a_{iw}x_w \leq 1, \quad i \in V, \tag{2.20}$$

$$\sum_{w\in\Omega} x_w \leq |K|, \tag{2.21}$$

$$x_w \in \{0,1\}, \quad w \in \Omega. \tag{2.22}$$

In this formulation, Ω is the set of all feasible workdays, d_w is the total distance of workday w and $g_w = \sum_{i\in V_w} g_i$, with V_w the set of customers in workday w, is the total gain of workday w. Since it is assumed that $g_i = 1, i \in V$, g_w is the number of customers in workday w. Coefficient a_{iw} is 1 if customer i is in workday w, 0 otherwise. A solution is thus a subset of workdays $\Omega' \subseteq \Omega$ that covers each customer at most once.

As the number of columns can be huge, the columns are progressively introduced into the master problem to obtain a series of restricted MPs (RMPs). The linear relaxation of each RMP (RLMP) is then solved with an LP solver. The dual variables associated with the optimal solution of a given RLMP are used to define a pricing subproblem that identifies workdays with a negative reduced cost, if any. These workdays are then added to the current RLMP to obtain the next RLMP, which is solved again to obtain new dual variables. This iterative procedure is repeated until no more workdays with negative reduced cost can be found. At this point, an optimal solution for the RLMP has been obtained.

2.3.2 Pricing subproblem

Each pricing subproblem is an elementary shortest path with resource constraints defined on an auxiliary graph $G^T = (V^T, A^T)$ which is constructed once at the

start of the algorithm [3]. This auxiliary graph is called a route graph, because each node corresponds to a feasible route in the original graph G. That is, the node set V^T is made of all feasible routes in G, plus two artificial nodes for the start and end of the workday (note that the number of feasible routes is limited due to the route duration constraint). Each route node r in this graph has two time windows which are derived from the time windows of the customers served in that route, namely, $[\underline{t}^r_0, \overline{t}^r_0]$ and $[\underline{t}^r_{n+1}, \overline{t}^r_{n+1}]$, where $\underline{t}^r_0$, $\overline{t}^r_0$, $\underline{t}^r_{n+1}$ and $\overline{t}^r_{n+1}$ are the earliest departure time, latest departure time, earliest arrival time and latest arrival time, respectively. The minimum route duration is thus $(\overline{t}^r_{n+1} - \overline{t}^r_0)$ because the waiting time is minimized by serving the route at the latest feasible time. The arc set A^T corresponds to feasible transitions between routes. In particular, given two route nodes r and s, arc $(r, s) \in A^T$ if routes r and s have no customers in common and route s can be served after route r, while satisfying the time window constraints. The arc cost c_{rs} is $c_{rs} = d_s - \alpha g_s$, where d_s is the total distance of route s and $g_s = \sum_{i \in V_s} g_i$, with V_s the set of customers in route s, is the total gain of route s (here, the number of customers in route s). There is also an arc from the artificial start node to every route node and from every route node to the artificial end node (in the latter case, the costs are set to 0).

We now formulate the pricing subproblem using the vector λ of dual variables associated with the packing constraints (2.20), the dual variable μ associated with constraint (2.21) and the reduced cost of arc (r, s) denoted $c'_{rs} = c_{rs} - \sum_{i \in V_s} \lambda_i$. In this formulation, the binary variables X_{rs} indicate if arc (r, s) is used or not in the route graph and the continuous variables T_r correspond to the departure time of route r. Finally, the nodes 0 and $n + 1$ are the artificial start and end nodes, respectively. In this formulation, constraints (2.24), (2.25) and (2.26) are the flow conservation constraints, while constraints (2.27) and (2.28) guarantee the feasibility of the time schedule.

We do not solve that formulation directly, but rather use the label correcting

algorithm of [9], denoted FDGG in the following, to solve the corresponding elementary shortest path problem. This is explained in the next section.

$$\text{Min} \sum_{(r,s)\in A^T} c'_{rs} X_{rs} - \mu \tag{2.23}$$

s.t.

$$\sum_{(r,h)\in A^T} X_{rh} - \sum_{(h,s)\in A^T} X_{hs} = 0, \quad h \in V^T, \tag{2.24}$$

$$\sum_{r\in V^T} X_{0r} = 1, \tag{2.25}$$

$$\sum_{r\in V^T} X_{r(n+1)} = 1, \tag{2.26}$$

$$T_r + \sigma^s + \left(\bar{t}^r_{n+1} - \bar{t}^r_0\right) - M(1 - X_{rs}) \leq T_s, \quad (r,s) \in A^T, \tag{2.27}$$

$$\underline{t}^r_0 \leq T_r \leq \bar{t}^r_0, \quad r \in V^T, \quad (r,s) \in A^T, \tag{2.28}$$

$$X_{rs} \in \{0,1\}, \quad (r,s) \in A^T, \tag{2.29}$$

$$T_r \geq 0, \quad r \in V^T. \tag{2.30}$$

2.3.3 Solving the pricing subproblem

Each pricing subproblem is an elementary shortest path problem with resource constraints from the artificial start node to the artificial end node in the route graph G^T. Due to the presence of negative cost cycles in G^T, the elementary condition is very important here. A path is characterized by the consumption of each resource, in addition to its cost. Accordingly, when different paths lead to the same node, it might well be that no path dominates, or is better than the others, with regard to the cost and the consumption of each resource. Thus, many different labels are maintained at each node (i.e., all non-dominated paths leading to that node).

In our case, the label L_p associated with a path p from the start node to a given route node r is the vector $(C_p, T_p^r, s_p, V_p^1, \ldots, V_p^{n^T})$, where C_p is the cost of path p, T_p^r is the time of beginning of service at route node r, s_p is the number of visited route nodes and $V_p^{r'}$, $r' = 1,\ldots, n^T$, is 1 if route node r' is visited, 0 otherwise. The $V_p^{r'}$ values provide a trace of previously visited route nodes along path p, thus allowing cycles to be detected. Note that when route node r is visited and its resource V_p^r is consumed, a similar resource is also consumed for every customer in route r. In this way, a workday made of routes with some common customers cannot be produced.

A path p is feasible with regard to the time constraints if $\underline{t}_0^r \leq T_p^r \leq \overline{t}_0^r$, for every route node r along the path. An arc (r, s) is thus added to a feasible partial path leading from the start node to route node r if it is possible for the vehicle to depart from the depot before the latest departure time of route node s. If this is the case, the cost of route s is incurred and the time consumed is the duration of route s, plus the setup time of route s, plus any waiting time before departure.

Based on the label vector presented above, the following (non strict) dominance relation is defined :

Dominance relation. If p and p' are two different paths from the start node to a given route node r with labels L_p and L'_p, respectively, then path p dominates p' if $C_p \leq C_{p'}$, $T_p^r \leq T_{p'}^r$, $s_p \leq s_{p'}$ and $V_p^{r'} \leq V_{p'}^{r'}$, $r' = 1, ..., n^T$.

That is, path p dominates p' if (i) it is not longer, (ii) it does not consume more time and (iii) every visited node in path p is also visited in path p'. Note that s_p is included in the label only to quickly filter out unpromising paths and speed up the computations. By eliminating paths through this dominance relation, only labels corresponding to non dominated elementary paths are kept and a solution to the problem is obtained at the end. That is, the sequence of nodes in a least

cost path from the start node to the end node in G^T corresponds to the sequence of routes in a vehicle's workday.

To accelerate the search, a heuristic version of the FDGG algorithm is first applied. That is, the condition $V_p^{r'} \leq V_{p'}^{r'}$, $r' = 1, ..., n^T$, is not checked in the dominance relation between paths p and p' (hence, path p' can be eliminated even if it is not really dominated by p). The exact algorithm is used only when the heuristic cannot find a workday with a negative reduced cost. The bidirectional approach of [60] has also been implemented although it did not significantly improve the performance of our algorithm. In the bidirectional approach, two different paths are generated : one leaves the start node and progresses forward, while the other leaves the end node and progresses backward. The two paths are joined together when they met.

2.3.4 The column generation algorithm

The column generation algorithm is embedded within a branch-and-price framework that generates a search tree, as explained in Section 2.4. Its main characteristics are as follow.

Initialization. At the root of the search tree, the RLMP is initialized with workdays made of a single customer visit. The number of columns thus corresponds to the number of customers. For the other nodes in the search tree, the algorithm initializes the RLMP with the set of columns in the last node considered, after removing columns that are infeasible due to branching.

Stabilization. One disadvantage of column generation is that many iterations are often needed to prove optimality, while the optimal value of the RLMP does not change significantly from one iteration to the next. This situation occurs, in particular, when the master problem shows some degree of degeneracy. Many

authors have thus proposed stabilization techniques to speed up convergence. Here, the interior point stabilization technique developed by [64] is used. This technique exploits solutions taken inside the optimal dual polyhedron, rather than extreme points, because their dual values tend to better approximate the true reduced costs. In practice, a number of dual solutions are generated inside the polyhedron and a convex combination of the dual values is used within the pricing subproblem. We refer the reader to [64] for further details.

2.4 The branch-and-price algorithm

Column generation is applied at every node of a search tree generated by the branch-and-price algorithm. Thus, this section describes our search and branching strategies.

2.4.1 Search strategy

The search tree is explored according to a best-first policy, where subproblems are ranked according to the value of their lower bound. Best-first was chosen over depth-first, as the latter yielded inferior results in preliminary experiments.

2.4.2 Branching strategy

For branching, we favored the master problem over the pricing subproblem, because the original graph is much smaller than the route graph. The branching schemes are considered in the following order : branching on the number of vehicles, branching on customers and branching on arcs. These strategies are all

compatible with the structure of the pricing subproblem. They are explained in the following.

Branching on the number of vehicles. First, we determine the number of vehicles $k = \sum_{w \in \Omega'} x_w$ in the optimal solution of the current RLMP, where $\Omega' \subseteq \Omega$ is the subset of columns in the RLMP. If this number is fractional, $\sum_{w \in \Omega'} x_w \leq \lfloor k \rfloor$ is imposed in the first branch and $\sum_{w \in \Omega'} x_w \geq \lfloor k+1 \rfloor$ in the second branch.

Branching on customers. We search for a customer i where $y_i = \sum_{w \in \Omega'} a_{iw} x_w$ is fractional. If several fractional customers exist, the one with associated value closest to 0.5 is selected. Two branches are created which respectively forbid ($y_i = 0$) and enforce ($y_i = 1$) customer i in the solution. In the first case, all columns or workdays that serve customer i are deleted from the RLMP, as well as all routes with this customer in the route graph. In the second case, the corresponding constraint in (20) is forced to 1. Branching on customers can make the RLMP infeasible because, once a substantial number of branching decisions has been made, many constraints of type (20) are forced to one. This situation is addressed by adding artificial variables z_i in (20) to obtain $z_i + \sum_{w \in \Omega'} a_{iw} x_w = 1, i \in V$. A very large cost is assigned to these variables to make sure that they are part of the solution only in case of infeasibility.

Branching on arcs. We also branch when the flow on any arc (i, j) is fractional. Two branches are created : one branch with $x_{ij} = 1$, where vertex j must be visited immediately after vertex i ; and the other branch with $x_{ij} = 0$, where it is forbidden to visit vertex j after vertex i. In the first case, all workdays in the RLMP, as well as all routes in the route graph, that contain arc (i, k) with $k \neq j$ or arc (k, j) with $k \neq i$ are deleted. Also, vertices i and j are forced in the RLMP by setting $y_i = 1$ and $y_j = 1$. In the second case, all workdays in the RLMP, as well as all routes in the route graph, where vertex j immediately follows vertex i, are deleted.

We also branch on two consecutive fractional arcs at once to reach integrality faster. Assuming that x_{ij} and x_{jk} are both fractional and close to 0.5, we impose $x_{ij}+x_{jk} \geq 1$ on one branch and $x_{ij}+x_{jk} = 0$ on the other branch. In the first case, an additional constraint in the RLMP states that the summation over workdays that contain arc (i, j) or arc (j, k) or both should be greater than or equal to 1. In the second case, all workdays in the RLMP, as well as all routes in the route graph, that contain one or both arcs (i, j) and (j, k) are deleted.

2.5 Computational Results

In the following, results obtained with our exact algorithm are reported. First, the test instances are presented. Then, the impact of the maximum route duration constraint is examined. Finally, the last subsection is devoted to the impact of time windows. The computing platform for all tests is an AMD Opteron 3.1 GHz with 16 GB of RAM, using ILOG CPLEX 10.0 to solve the RLMPs.

2.5.1 Test instances

Solomon's 100-customer Euclidean VRPTW instances are used to test our algorithm. In these instances, the travel time and the Euclidean distance between two customer locations are the same and this value is truncated to two decimal places. There are six different classes of instances depending on the geographic location of the customers (R : random ; C : clustered ; RC : mixed) and width of the scheduling horizon (1 : short horizon ; 2 : long horizon). In this work, instances of type 1 are discarded due to the short horizon that does not allow a significant number of routes to be sequenced to form a workday. Results are thus reported for R2 (11 instances), C2 (8 instances) and RC2 (8 instances), for a total of 27

instances. Due to the limitations of our exact approach, the computational study focuses on instances obtained by taking only the first 25 or 40 customers from each original instance. However, it should be noted that the branch-and-price algorithm can solve a few instances of size 50. They are the largest instances that can currently be addressed with this algorithm.

2.5.2 Impact of route duration

Solomon's VRPTW test instances are modified to fit our problem. In particular, a value t_{max} to limit route duration is needed. This value was first set to 75 in the case of R2 and RC2, and 220 in the case of C2. The value is larger for C2 because the service or dwell time at each customer is 90, as opposed to 10 for R2 and RC2. Larger t_{max} values are also considered to allow more customers per route, namely, 100 for R2 and RC2 and 250 for C2. Also, constant β, which is used to weigh the vehicle loading time with regard to the sum of service times, is set to 0.2. We chose a value which seems realistic to us. Some tests with $\beta = 0$ have shown that this constant does not impact the algorithmic performance. Finally, a gain of 1 is associated with each customer and weighted by an arbitrarily large constant to maximize first the number of served customers, and then minimize the total distance.

All results are recorded at the root node and at the end of the branch-and-price algorithm (due either to timeout or optimality). These results are shown in Tables 2.1 and 2.2. In these tables, a particular instance is identified by its class and its index followed by a dot and the number of customers considered. For example rc202.25 is the second instance of class RC2, where only the first 25 customers are considered. In these tables, column *Problem* is the identifier of the problem instance, *Gap* is the gap in % between the value of the linear relaxation at the root and the optimal value, *Cols* is the total number of columns generated

during the branch-and-price algorithm, *Iter* is the total number of RLMPs solved by CPLEX, *Nodes* is the number of nodes explored in the search tree, *Dist.* is the total distance, *% Cust.* is the percentage of served customers, *Routes per day* is the average number of routes in a workday, *Cust. per route* is the average number of customers in a route and *CPU* is the computation time in seconds.

Table 2.1 shows the results obtained on the 25-node and 40-node instances with 2 vehicles using the smaller t_{max} values 75 (R2 and RC2) and 220 (C2). We can observe that, out of 27 instances, 15 instances of size 25 are solved within the alloted time of 72 hours. Also, all customers are served by the vehicles. The CPU times vary widely and range from a few seconds to many hours. When the number of customers increases to 40, 7 instances out of 27 are solved to optimality, with a significant increase in CPU time. In some cases, only a subset of customers is served. It is worth noting that an increase in the number of vehicles to allow all customers to be served does not make the instance harder to solve, because the route graph remains the same.

Table 2.2 shows similar results, but for the larger t_{max} values 100 (R2 and RC2) and 250 (C2). This relatively modest increase in the t_{max} values significantly impacts the branch-and-price algorithm. Due to the larger number of customers who can be accomodated in each route, the complexity quickly increases. Hence, a smaller number of instances are now solved within the alloted time, namely, 14 instances of size 25 and 4 instances of size 40. This is to be compared with 15 instances of size 25 and 7 instances of size 40 for the smaller t_{max} values. Also, important increases in computation times are generally observed. There are, however, notable exceptions : rc206.25, r201.25, c201.25 and c201.40 are solved faster with the larger t_{max} values, while rc202.25 c205.25 and c206.25 are solved with the larger t_{max} values, but not with the smaller ones. As expected, the average number of customers per route also increases with larger t_{max} values.

Instance	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
rc201.25	0.14%	1474	364	22	988.05	100%	6.0	2.08	3.1
rc205.25	2.19%	1225	167	22	840.35	100%	5.0	2.50	28.8
rc206.25	2.28%	35363	7041	1326	761.03	100%	4.5	2.77	7156.8
r201.25	0.43%	7382	1909	166	762.43	100%	6.0	2.08	68.3
r202.25	0.00%	1427	31	0	645.78	100%	4.5	2.77	205.2
r203.25	0.00%	2067	31	0	621.97	100%	4.5	2.77	1333.2
r204.25	0.32%	26043	1079	260	579.68	100%	4.0	3.12	30983.3
r205.25	0.50%	9725	1461	218	634.09	100%	4.5	2.77	354.1
r206.25	0.00%	1509	19	0	596.74	100%	4.0	3.12	318.4
r207.25	0.16%	4059	76	26	585.74	100%	4.0	3.12	2853.5
r208.25	0.32%	10243	334	98	579.68	100%	4.0	3.12	9270.3
r209.25	0.30%	3241	233	30	602.39	100%	4.0	3.12	262.6
r210.25	1.33%	14017	1299	214	636.15	100%	4.5	2.77	5094.1
r211.25	0.58%	15131	1005	172	575.91	100%	4.0	3.12	5648.6
c201.25	0.90%	49088	11793	1134	659.02	100%	5.5	2.27	40361.2
rc201.40	1.87%	1463	298	86	1292.16	77.5%	7.5	2.06	14.6
rc202.40	0.87%	10746	951	134	1457.89	92.5%	8.0	2.31	6823.2
rc205.40	2.28%	5507	951	98	1272.12	85%	7.0	2.42	1904.2
r201.40	0.79%	6221	1224	78	1130.59	95%	8.5	2.23	2979.5
r205.40	0.69%	43115	5264	650	1017.84	100%	7.0	2.85	244494.0
c201.40	0.59%	57812	11574	618	1168.83	100%	9.0	2.22	19978.9
c208.40	0.36%	22169	969	68	1071.99	100%	8.0	2.50	3221.7

TAB. 2.1 – Instances solved to optimality with t_{max} values 75 and 220

Instance	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
rc201.25	1.16%	1280	215	32	849.33	100%	4.5	2.77	16.06
rc202.25	0.00%	1421	34	2	679.86	100%	4.0	3.12	1096.3
rc205.25	0.06%	759	57	6	702.49	100%	4.0	3.12	262.8
rc206.25	0.03%	1248	133	14	604.12	100%	3.5	3.57	222.7
r201.25	0.24%	1257	200	18	698.18	100%	4.0	3.12	43.6
r202.25	0.06%	4156	214	38	617.53	100%	4.0	3.12	25249.9
r203.25	0.00%	2135	26	0	577.74	100%	4.0	3.12	75729.3
r205.25	0.28%	1901	163	34	559.14	100%	4.0	3.12	1202.3
r206.25	0.00%	1695	31	2	523.64	100%	3.5	3.57	28498.1
r209.25	0.20%	1998	149	46	517.69	100%	3.0	4.16	11173.9
r210.25	0.00%	1230	25	6	547.23	100%	3.5	3.57	26690.2
c201.25	0.00%	683	42	6	540.9	100%	4.5	2.77	1.3
c205.25	0.52%	5361	933	126	529.94	100%	4.5	2.77	116.6
c206.25	0.66%	20092	3888	802	527.84	100%	4.5	2.77	1987.2
rc201.40	0.33%	782	96	8	1157.48	85%	6.5	2.61	77.8
rc205.40	1.41%	1721	100	6	1195.32	95%	6.5	2.92	4733.3
r201.40	1.47%	15015	3120	250	1075.05	97.5%	7.0	2.85	127424.0
c201.40	0.13%	5730	1191	62	966.7	100%	7.5	2.66	659.2

TAB. 2.2 – Instances solved to optimality with t_{max} values 100 and 250

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
rc201.25	0.27 %	192	34	14	967.92	88%	6.0	1.83	0.1
rc202.25	3.43%	6996	1877	774	961.62	96%	6.0	2.00	228.1
rc203.25	2.93%	7871	2013	562	752.31	96%	4.5	2.66	3091.4
rc205.25	2.99%	944	218	50	974.97	96%	6.0	2.00	5.4
rc206.25	0.43%	1703	444	62	978.13	100%	6.0	2.08	4.6
rc207.25	5.14%	13989	3119	950	819.33	96%	5.0	2.40	418.4
r201.25	0.49%	577	110	10	772.30	92%	5.5	2.09	1.0
r202.25	0.00%	1145	108	6	694.57	96%	5.0	2.40	127.0
r203.25	0.76%	4160	676	118	657.95	96%	4.5	2.66	3239.5
r204.25	0.46%	17999	3187	738	574.38	96%	4.0	3.00	37195.3
r205.25	0.43%	4930	1326	158	762.43	100%	6.0	2.08	60.1
r206.25	0.41%	9303	1795	362	684.89	100%	5.0	2.50	2058.7
r207.25	0.31%	4428	686	82	654.52	100%	4.5	2.77	4138.9
r208.25	1.41%	83114	10107	2008	609.81	100%	4.5	2.77	106130.0
r209.25	0.62%	3332	555	66	688.99	100%	5.0	2.50	52.5
r210.25	0.00%	918	71	2	703.62	100%	5.0	2.50	121.4
r211.25	0.00%	1150	57	10	622.69	100%	4.0	3.12	42.9
c201.25	0.94%	2645	713	98	678.01	96%	6.0	2.00	5.1
c202.25	0.88%	13860	4023	606	672.71	100%	6.0	2.08	782.8
rc201.40	0%	163	16	2	1182.67	65%	7.0	1.85	0.2
rc202.40	2.33%	5859	1396	530	1271.87	75%	7.5	2.00	1439.4
rc205.40	0.03%	509	47	10	1146.49	72.5%	6.5	2.23	6.0
rc206.40	0.61%	538	70	22	1197.37	75%	7.0	2.14	3.5
rc207.40	2.61%	3062	461	138	1082.38	77.5%	6.0	2.58	233.1
r201.40	0.79%	4343	869	94	1016.34	82.5%	8.5	2.06	145.9
r202.40	0.67%	11975	2467	242	935.52	87.5%	7.0	2.50	237127.0
r205.40	1.54%	9976	1808	222	1054.75	92.5%	8.0	2.31	4964.5
r209.40	0.69%	10271	996	68	970.49	95%	7.5	2.53	25113.4
c201.40	1.10%	3173	700	42	1049.34	92.5%	8.5	2.17	31.5
c202.40	1.10%	7808	851	62	1102.69	97.5%	8.0	2.37	21600.2
c205.40	0.67%	20021	5330	346	1171.58	100%	9.0	2.22	1461.6
c206.40	0.41%	40951	6816	310	1125.2	100%	8.5	2.35	6954.9
c208.40	0.41%	39830	4969	240	1106.79	100%	8.5	2.35	10338.5

Tab. 2.3 – Instances solved to optimality with reduced time windows

2.5.3 Impact of time windows

Here, we first consider new test instances obtained by reducing in half the width of the time windows to obtain more constrained instances. The results for the instances with reduced time windows are shown in Table 2.3 using t_{max} values 75 (R2 and RC2) and 220 (C2). Clearly, these new instances are easier to handle by the branch-and-price algorithm. Now, 19 instances with 25 customers and 14 instances with 40 customers can be solved, as compared with 17 instances and 8 instances, respectively, with the original time windows. Furthermore, the CPU times are often much faster.

We also removed the time windows to obtain less constrained instances. Given that all instances in the same class differ only by the time windows, a single instance without time windows is obtained for each one of the three classes R2, C2 and RC2. The results for the instance with 25 customers in class R2 is shown in Table 2.4. The two instances with 25 customers in classes C2 and RC2, as well as the three instances with 40 customers, could not be solved within the alloted time. This test shows that the absence of time windows leads to instances that are much more difficult to solve.

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
r201.25	0.74%	53053	1722	554	574.23	100%	4.0	3.12	21107.2

TAB. 2.4 – Instance solved to optimality with no time windows

2.5.4 Route graphs

Tables 2.5 and 2.6 report the average number of nodes, average number of arcs and density of the route graphs for the original RC2 instances, as well as those with reduced time windows, with 25 and 40 customers, respectively, using

t_{max} values 75 and 220. These results include the instances that are not solved to optimality.

Instances rc204 and rc208 are associated with the largest route graphs, that is, the largest number of nodes and arcs, as well as the largest density. Not surprisingly, these instances cannot be solved by the branch-and-price algorithm. In the case of rc208, the size of the route graph significantly decreases when the time windows are cut in half. The decrease is not as important in the case of rc204. In this instance, the width of almost every time window is very large and extends over the entire scheduling horizon (or close to it). Thus, reducing the width of each time window in half still leaves a lot of flexibility.

Instance	Original			Reduced TW		
	#Nodes	#Arcs	Density	#Nodes	#Arcs	Density
rc201.25	79	2425	38.37	55	1157	37.56
rc202.25	290	33698	39.93	231	16765	31.28
rc203.25	471	110437	49.67	362	56522	43.01
rc204.25	583	293159	86.10	461	103006	48.36
rc205.25	253	22600	35.16	148	7455	33.80
rc206.25	165	11136	40.65	87	2834	37.01
rc207.25	385	59181	39.82	157	8327	33.56
rc208.25	538	185962	64.12	233	21687	39.77

Tab. 2.5 – Route graphs for RC2 instances with 25 customers

2.6 Conclusion

In this paper, the first exact branch-and-price algorithm for solving the vehicle routing problem with time windows and multiple use of vehicles was proposed. Although the algorithm is limited by the problem size and some characteristics

Instance	Original			Reduced TW		
	#Nodes	#Arcs	Density	#Nodes	#Arcs	Density
rc201.40	113	5327	41.35	82	2529	37.15
rc202.40	365	61693	46.18	282	30120	37.74
rc203.40	559	173591	55.45	428	90211	49.13
rc204.40	733	350231	65.09	641	244876	59.50
rc205.40	311	39412	40.61	185	12381	35.98
rc206.40	216	21527	45.92	122	5930	39.51
rc207.40	519	130345	48.29	224	21647	42.95
rc208.40	673	326307	71.93	299	43600	48.60

Tab. 2.6 – Route graphs for RC2 instances with 40 customers

of the problem (like the route duration constraint), it can still routinely solve instances with 25 customers and a few instances with up to 50 customers. Clearly, a heuristic approach remains a viable alternative for larger instances and our current research efforts are aimed in that direction.

Chapitre 3

An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips

Résumé. Dans le problème de tournées avec réutilisation des véhicules, chaque véhicule peut effectuer plusieurs tournées durant une journée de travail. Ce problème est relié à diverses applications où la durée de chaque route est limitée, par exemple, lorsque les biens à transporter sont des denrées périssables. Nous supposons qu'on dispose d'une flotte fixe de véhicules et qu'il n'est pas toujours possible de desservir tous les clients. L'objectif est donc de maximiser le nombre de clients à desservir, et pour un même nombre de clients, de minimiser la distance totale parcourue. Une recherche adaptative à voisinage large est proposée pour résoudre le problème. Plusieurs opérateurs de destruction et de construction ont été définis, qui tiennent compte de la nature hiérarchique du problème, en opérant d'abord au niveau de la journée de travail des véhicules, puis au niveau des routes et

enfin au niveau des clients. Des résultats expérimentaux obtenus en utilisant des instances connues démontrent les avantages de l'approche multi-niveaux.

An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips

Nabila Azi
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Michel Gendreau
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Jean-Yves Potvin
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Cet article a été soumis pour publication dans *Computers & Operations Research*

Abstract. The vehicle routing problem with multiple trips consists in determining the routing of a fleet of vehicles where each vehicle can perform multiple routes during its workday. This problem is relevant in applications where the duration of each route is limited, for example when perishable goods are transported. In this work, we assume that a fixed-size fleet of vehicles is available and that it might not be possible to serve all customer requests, due to time constraints. Accordingly, the objective is first to maximize the number of served customers and then, to minimize the total distance traveled by the vehicles. An adaptive large neighborhood search, exploiting the ruin-and-recreate principle, is proposed for solving this problem. The various destruction and reconstruction operators take advantage of the hierarchical nature of the problem by working either at the customer, route or workday level. Computational results on Euclidean instances, derived from well-known benchmark instances, demonstrate the benefits of this multi-level approach.

Keywords : Vehicle routing, multiple trips, adaptive large neighborhood search, ruin-and-recreate, multi-level.

3.1 Introduction

Studies on the classical vehicle routing problem (VRP) and its many variants represent a significant fraction of the operations research literature [73]. However, most studies assume that a vehicle can do no more than a single route during a given scheduling period (typically, a day). In this work, we consider a vehicle routing problem where a vehicle is allowed to perform multiple routes (VRPM) starting from and ending at a central depot. Furthermore, each customer has an associated time window for service (VRPMTW). The vehicle can arrive at the customer location before the lower bound of the time window, and must wait up to this lower bound, but is not allowed to arrive after the upper bound. These problems arise, for example, in e-grocery services where perishable goods are delivered to customers who must be on-site. This application leads to multiple short vehicle routes, where the last customer in each route must be visited within a given time limit from the start of the route.

One of the first study on the VRPM is found in Taillard et al. [70]. First, different solutions to the classical VRP are produced with a tabu search heuristic. Then, the routes obtained are combined to form workdays for the vehicles by heuristically solving a bin packing problem, an idea previously found in [24]. Due to the heuristic nature of the bin packing solutions, one or more workdays might extend over the scheduling period, leading to overtime which is penalized in the objective. In [10], the authors report a tabu search heuristic for solving a real-world application where additional characteristics are taken into account, like an heterogeneous fleet of vehicles with limited access restrictions, maximum legal driving time per day (penalized, if there is overtime), etc... In a later work [11], the same authors apply a streamlined version of their tabu search heuristic on the vehicle routing instances with multiple trips used in [70]. The results show that their algorithm is competitive with Taillard et al.'s method. A multi-phase heu-

ristic, similar in spirit to [70], but using a more involved bin packing heuristic, is proposed in [51]. In this work, a savings-based construction method and a (giant) tour partitioning method are first applied to generate a pool of VRP solutions. The routes obtained are then used to produce solutions to the VRPM through the bin packing heuristic. It is worth noting that different exchange heuristics are also applied to improve both the VRP and VRPM solutions. A hybrid genetic algorithm for the VRPM is reported in [52], where the genetic operators are specifically designed for multi-trip vehicle routing solutions. A problem-solving method based on an adaptive memory made of elite solutions [62], a concept related to the population-based approach of genetic algorithms, is also reported in [50]. In [1], a site-dependent periodic vehicle routing problem is described where a vehicle can perform more than one route per day over an horizon of a few days. The problem is addressed with a tabu search heuristic.

Recently, an adaptive guidance algorithm was proposed for a variant of the VRPMTW. This variant stems from a real-world distribution problem where different types of pair-wise incompatible commodities must be delivered to customers [7]. A decomposition approach generates simpler subproblems which are then solved with specific heuristics. Two adaptive guidance mechanisms are defined : one is based on penalization of critical time intervals (i.e., intervals where many routes are strongly active) and improvement of routes with critical commodities (i.e., commodities that do not seem to be well packed across multiple routes). Finally, an exact approach for the VRPMTW is reported in [5].

Here, the pervasive issue of a (possible) inability of the fixed-size fleet of vehicles to accommodate all customers, due to the finite time horizon, is addressed by visiting only a subset of customers rather than by allowing overtime. This approach is required to account for the time window constraints that may preclude the existence of any feasible solution, even by considering overtime. An Adaptive Large Neighborhood Search (ALNS) [53, 63] is proposed to address the

VRPMTW. The ALNS is designed to account for the hierarchical nature of the problem through the application of operators that modify the current solution at the customer, route and workday levels. It is empirically shown that this multi-level approach leads to much better solutions than the classical customer-based approach. The rest of the paper is organized as follows. The problem is first precisely defined in section 4.2. Then, our algorithm is described in section 3.3. Computational results are reported in section 3.4 and a conclusion follows.

3.2 Problem definition

Our problem can be stated as follows. We have a directed graph $G = (V, A)$, where $V = \{0, 1, 2, ..., n\}$ is the set of customer vertices with the depot 0 and where A is the arc set. With each customer $i \in V - \{0\}$ is associated a gain g_i, a demand q_i, a service or dwell time s_i and a time window $[a_i, b_i]$, where a_i and b_i are the earliest and latest time, respectively, to begin the service (with $a_0 = 0$ and $b_0 = \infty$). Thus, a vehicle has to wait if it arrives at customer i before a_i. With each arc $(i, j) \in A$ is associated a distance d_{ij} and a travel time t_{ij} (in this work, distances and travel times are the same). We also have a set $K = 1, 2, ..., m$ of vehicles, each of capacity Q, to deliver goods from the depot to customers. The duration of each route is limited by forcing the last customer to be served within t_{max} time units of the route start time. This restriction leads to short routes that must be combined and sequenced to form vehicle workdays. Also, a setup time for loading the vehicle, noted σ_r, is associated with each route r in the solution. Here, the setup time is proportional to the sum of service times over all customers in the route.

The objective considered is hierarchical : first, the number of served customers is maximized (by maximizing the total gain, assuming a gain of 1 for every cus-

tomer) ; second, for the same number of customers, the total distance traveled by the vehicles is minimized. A complete mathematical description of this problem can be found in [5].

3.3 Problem-solving methodology

ALNS extends the large neighborhood search framework of Shaw [66], a problem-solving approach which can also be related to the ruin-and-recreate principle [65]. The basic idea is to search for a better solution at each iteration by destroying a part of the current solution and by reconstructing it in a different way. When solving VRPs, a new solution is typically obtained by first removing a number of customers and then by reinserting these customers into the solution. In general, a number of destruction and reconstruction operators are available and a destruction-reconstruction pair is randomly chosen at each iteration. In the adaptive extension, a weight is associated with each operator and the selection probability of an operator is related to its weight, which is adjusted during the search based on its past successes.

The problem structure, where a vehicle workday is made of a sequence of trips and where each trip is made of a sequence of customers, is exploited by applying destruction operators at the workday, route and customer levels, in this order. This approach is described in pseudo-code in Procedure 3.3.1. The idea is to go from gross (high-level) to fine (low-level) refinements. In this algorithm, an initial feasible solution s is first constructed. Then, a destruction and a reconstruction operator are probabilistically chosen based on their current weights. The destruction operators are first selected at the workday level, then at the route level and finally at the customer level, where each level is explored for a number of consecutive iterations. At each iteration, a new solution is obtained by applying

the destruction operator followed by the reconstruction operator on the current solution s. The new solution s' is then submitted to an acceptance rule. If accepted, the new solution becomes the current solution, otherwise the current solution does not change. After exploring a given level, the weights associated with the applied operators are adjusted. This is repeated until a termination criterion is met and the best solution found $s*$ is returned. In the following, each component of this algorithmic framework will be explained in detail.

Procedure 3.3.1 ALNS

---

1. construct a feasible solution s ;

2. $s* \leftarrow s$;

3. initialize weights ;

4. **while** the stopping criterion is not met **do**

4.1 for L = workday, route, customer **do**

4.1.1 for I iterations **do**

a. probabilistically select a destruction operator at level L and a reconstruction operator based on their current weights ;

b. apply the destruction and reconstruction operators to s to obtain s' ;

c. if s' satisfies the acceptance criterion **then**

- $s \leftarrow s'$;

- **if** s' is better than $s*$ **then** $s* \leftarrow s'$;

4.1.2 adjust weights ;

5. return $s*$.

3.3.1 Construction of the initial solution

An insertion heuristic, where all routes and workdays grow in parallel, is used to obtain an initial solution, see Procedure 3.3.2. Based on a given ordering, each

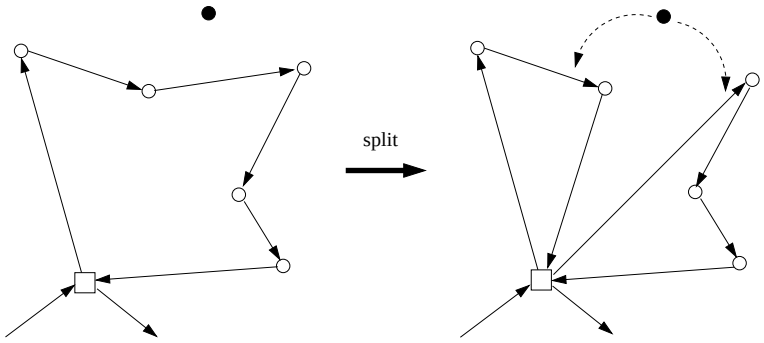

FIG. 3.1 – Split

customer is considered in turn and inserted at its best feasible insertion place over every route in every workday, including a new (empty) workday if one is still available. In this work, the best insertion place corresponds to the smallest detour in distance, where the detour is $d_{ji} + d_{il} - d_{jl}$ for the insertion of customer i between j and l. If there is no feasible insertion place for customer i, then each route is considered in turn and split into two subroutes, with an additional copy of the depot between the two subroutes. This is illustrated in Figure 3.1 where a route in the workday of a single vehicle is shown. In this figure, the square stands for the depot and the black circles for the customers to be inserted. Once the route is split, the insertion of the customer can take place in any of the two new routes in the vehicle's workday. Each route is split in every possible way (i.e., at every customer location along the route) to find the best insertion place. If there is still no feasible insertion place, then customer i is put in a list of (temporarily) unserved customers. The split procedure proved to be particularly useful when the infeasibility is due to the t_{max} constraint.

During this insertion procedure, the vehicle schedule based on the latest feasible departure times from the depot is considered to reduce as much as possible route durations. Let us assume that route r is the last route in a vehicle workday. This route is described by the sequence $(0_d^r = i_0^r, i_1^r, i_2^r, ..., i_{n_r}^r, i_{n_r+1}^r = 0_e^r)$, where

Procedure 3.3.2 Construction heuristic

1. $L \leftarrow \{1, 2, ..., n\}$;

2. while $(L \neq \emptyset)$ **do**

2.1 randomly select customer $i \in L$;

2.2 insert customer i at its best feasible insertion place (minimum detour) in the solution ;

2.3 if there is no feasible insertion place in the solution **then**

2.3.1 for each route k in the solution apply *split(i, k)* ;

2.4 $L \leftarrow L - \{i\}$.

0_d^r and 0_e^r stand for the departure from and return to the depot, respectively, and n_r is the number of customers in the route.

To determine the latest schedule of that route, denoted by the latest feasible time to begin service $\bar{t}_{i_j^r}$ at each customer i_j^r, $j = 1, ..., n_r$, a backward sweep of the route is first applied from $i_{n_r+1}^r = 0_e^r$ to $i_0^r = 0_d^r$ as follows :

$$\bar{t}_{i_{n_r+1}^r} \leftarrow b_{i_{n_r+1}^r},$$

$$\bar{t}_{i_j^r} \leftarrow min\{\bar{t}_{i_{j+1}^r} - t_{i_j^r i_{j+1}^r} - s_{i_j^r},\ b_{i_j^r}\},\ j = i_{n_r}^r, ..., i_0^r.$$

Once the latest departure time from the depot $\bar{t}_{i_0^r}$ is obtained, a forward sweep is applied from $i_0^r = 0_d^r$ to $i_{n_r+1}^r = 0_e^r$ to reset the $\bar{t}_{i_j^r}$ values and get the latest feasible schedule :

$$\bar{t}_{i_j^r} \leftarrow max\{\bar{t}_{i_{j-1}^r} + t_{i_{j-1}^r i_j^r} + s_{i_{j-1}^r},\ a_{i_j^r}\},\ j = i_1^r, ..., i_{n_r+1}^r.$$

The total route duration is $\bar{t}_{i_{n_r+1}^r} - \bar{t}_{i_0^r}$, which is also the minimum duration, because the waiting time is minimized by serving the route at the latest feasible time. The whole procedure is then applied to the second-to-last route $r - 1$ by setting

$$\bar{t}_{i_{n_{r-1}+1}^{r-1}} \leftarrow \bar{t}_{i_0^r} - \sigma_r.$$

This is repeated until the first route is done.

3.3.2 Destruction operators

Different destruction operators are defined at the customer, route and workday level. They are described in the following.

Customer level

These operators remove a number n_1 of individual customers from the current solution.

Random customer removal. This is a very simple approach where the customers are chosen at random.

Related customer removal. This operator is inspired from the one described by Shaw in [66]. The main difference is that the proximity measure between two customers is based on both spatial and temporal characteristics. This operator can be described as follows (see Procedure 3.3.3).

Starting with a randomly chosen request, which starts the whole procedure, the removal of the next request is (probabilistically) biased toward those that are close to one of the previously removed requests. The proximity between two customers is based on a metric that accounts for both the geographical distance and the absolute difference between the time of beginning of service at both customer locations, weighted by parameters α and β, respectively. Parameter d in step 3.5 controls the intensity of the bias. Namely, a high value for parameter d strongly favors the removal of requests that are close to previously removed requests (and conversely). Based on preliminary experiments, this parameter was

Procedure 3.3.3 Related customer removal

1. randomly select a customer j and remove it from the solution ;

2. $L \leftarrow \{j\}$;

3. **while** $\mid L \mid < n_1$ **do**

3.1 $i \leftarrow$ randomly select a customer in L ;

3.2 **for** each customer j in the solution **do**

3.2.1 $z_{ij} \leftarrow \alpha \cdot |t_i - t_j| + \beta \cdot d_{ij}$;

3.3 sort the z_{ij}'s in non decreasing order and put them in list B ;

3.4 choose a random number x between 0 and 1 ;

3.5 $pos \leftarrow \lceil \mid B \mid \cdot x^d \rceil$;

3.6 select customer j associated with the z_{ij} value at position pos in B and remove it from the solution ;

3.7 $L \leftarrow L \cup \{j\}$;

4. return L

set to 6.

Route level

These operators remove a number n_2 of individual routes from the current solution.

Random route removal. This is a very simple approach where the routes are chosen at random.

Related route removal. This is an adaptation of the corresponding customer-based operator which is applied at the route level. Accordingly, the algorithmic framework is very similar to the one found in Procedure 3.3.3. First, a route is randomly chosen. Then, the removal of the next route is (probabilistically) biased toward those that are close to one of the previously removed routes.

Procedure 3.3.4 Related route removal

1. randomly select a route r_j and remove it from the solution ;

2. $L \leftarrow \{r_j\}$;

3. while $\mid L \mid < n_2$ **do**

3.1 $r_i \leftarrow$ randomly select a route in L ;

3.2 for each route r_j in the solution **do**

3.2.1 $z_{ij} \leftarrow$ compute the proximity measure between r_i and r_j ;

3.3 sort the z_{ij}'s in non decreasing order and put them in list B ;

3.4 choose a random number x between 0 and 1 ;

3.5 $pos \leftarrow \lceil \mid B \mid \cdot x^d \rceil$;

3.6 select route r_j associated with the z_{ij} value at position pos in B and remove it from the solution ;

3.7 $L \leftarrow L \cup \{r_j\}$;

4. return L.

Two different proximity measures between route r_j and some previously removed route r_i have been considered, where it is assumed that a route is defined by its set of customers.

1. GC : $z_{ij} \leftarrow d_{(g_{r_i} g_{r_j})}$, where g_{r_i} amd g_{r_j} are the gravity centers of routes r_i and r_j, respectively.
2. $MinD$: $z_{ij} \leftarrow \min_{k \in r_i, l \in r_j} d_{kl}$.

In the first case, the proximity is measured by the distance between the gravity centers of routes r_i and r_j, where the latter is defined as the average location over all customer locations in the route. In the second case, the proximity is measured by the smallest distance between any pair of customers taken from routes r_i and r_j.

Workday level

A single operator is defined at the workday level. It simply removes n_3 randomly chosen workdays from the solution.

3.3.3 Insertion operators

After the application of a destruction operator, the customers that are not part of the solution, either because they have just been removed or because there was no feasible insertion place for them, are considered for reinsertion. Two different insertion heuristics have been defined for this purpose.

Least-cost heuristic

This is the insertion heuristic used for constructing an initial solution (see Procedure 3.3.2), except that it is applied only to customers that are not part of the solution. Accordingly, the selection of the next customer is random and its insertion takes place at the feasible location with minimum detour.

Regret-based heuristic

A second insertion heuristic has been devised to alleviate the myopic behavior of the least-cost heuristic. This is done by defining a reinsertion order based on a regret measure. For a given customer, the 2-regret heuristic computes the minimum feasible detour in each workday. Then, it considers the difference between the detour in the second best and best workday. If this difference is large, the corresponding customer gets high priority for reinsertion because a large cost is incurred if its best workday becomes infeasible (due to the insertion of other

customers). A generalized variant considers the minimum detour in each workday and sums up the differences, over all workdays, between the minimum detour in the workday and the overall minimum detour.

More formally, let us assume that the minimum detour when customer i is inserted in workday k is $detour_{ik}$ and that the overall minimum detour is obtained when customer i is inserted in workday k^*. Then, the generalized regret measure of customer i is :

$$gen\_regret_i = \sum_{k=1,\ldots,m} (detour_{ik} - detour_{ik^*}) .$$

At each iteration, the customer with the maximum regret measure is selected for reinsertion at the feasible insertion place with minimum detour.

3.3.4 Acceptance criterion

The criterion for accepting or rejecting a new solution is the one used in simulated annealing [41]. That is, the new solution s' is accepted over the current solution s if s' is better than s. Otherwise, it is accepted with probability

$$e^{-\frac{f(s')-f(s)}{T}}$$

where T is the temperature parameter and f is the objective function. Starting from some initial value, the temperature is lowered from one iteration to the next by setting $T \leftarrow c \cdot T$. Clearly, the probability of accepting a non-improving solution diminishes with the value of T, as the algorithm unfolds. This behavior allows the algorithm to progressively settle in a (hopefully) good local optimum. In our experiments, the starting temperature was set to $1.05 \cdot f(s_0)$, where s_0 is the initial solution, and c to 0.99975, as suggested in [63].

3.3.5 Adaptive mechanism

The ALNS applies a removal and an insertion operator at each iteration on the current solution. The adaptive mechanism is aimed at choosing the removal and insertion operators in a way that accounts for their previous outcomes. A weight is associated with each operator for this purpose. Let us assume that we have h insertion operators, each with a weight w_j, $j = 1, ..., h$. The insertion operator i is then selected with probability

$$\frac{w_i}{\sum_{j=1}^{h} w_j}, i = 1, ..., h.$$

That is, the probability of selecting a given operator increases with its weight. Starting with a unit weight for each insertion operator, the weights are updated after a number of consecutive iterations (200 iterations in our implementation), called a segment. The weights at the start of a given segment sg are based on those used in the previous segment $sg - 1$ and are computed as follows :

$$w_i^{sg} = \gamma \cdot w_i^{sg-1} + (1 - \gamma) \cdot \pi_i^{sg-1},$$

where γ has a value between 0 and 1 and π_i^{sg-1} is the score of operator i at the end of segment $sg - 1$. This score, reset to zero at the beginning of each segment, is incremented when insertion operator i is used at a given iteration t to produce a new solution. More precisely, the new score at iteration $t + 1$ becomes

$$\pi_i^{t+1} = \pi_i^t + \begin{cases} \sigma_1 & \text{if a new best solution has been produced,} \\ \sigma_2 & \text{if the solution produced is better than the current solution,} \\ \sigma_3 & \text{if the solution produced is accepted,} \\ & \text{but is worse than the current solution,} \end{cases}$$

where σ_1, σ_2 and σ_3 are parameters.

Parameter γ controls the inertia in the weight update equation. When γ is

close to 1, the history prevails and the weights do not change much. Conversely, when γ is close to 0, the update is driven by the most recent score.

The same approach is also used to update the weights of the removal operators at each level.

3.3.6 Termination criterion

The termination criterion is based on a fixed number of 24,000 iterations. This number has been chosen to allow convergence even on the largest test instances.

3.4 Computational Results

Solomon's 100-customer Euclidean VRPTW instances [68], as well as the 200-, 400-, 600-, 800- and 1,000-customer instances of Gehring and Homberger [26], were used to test our algorithm. In these Euclidean instances, the travel time between two customer locations is the same as the Euclidean distance. There are six different classes of instances depending on the location of the customers (R : random ; C : clustered ; RC : mixed) and width of the scheduling horizon (1 : short horizon ; 2 : long horizon). In this study, instances of type 1 have been discarded due to the short horizon that does not allow a significant number of routes to be sequenced to form a workday. Results are thus reported for R2 (11 instances), C2 (8 instances) and RC2 (8 instances). All tests were run on an AMD Opteron 3.1 GHz with 16 GB of RAM.

Solomon's instances, as well as Gehring and Homberger's VRPTW instances had to be modified to fit our problem. In particular, t_{max} was set to 100 to generate multiple routes for each vehicle. The customer coordinates of Gehring and

Homberger's instances were also normalized to fit within a 100 X 100 Euclidean square, as in Solomon's instances. Furthermore, the service or dwell time at each customer was set to 10 in all instances. The number of vehicles was set to 3 for the 100-customer instances, and this number was increased to 6, 12, 18, 24 and 30 for the 200-, 400-, 600-, 800- and 1,000-customer instances, respectively, to obtain instances with approximately the same degree of difficulty.

In the following, some parameter sensitivity results are presented. Then, the final results on the whole test set are reported. A comparison with known optimal solutions on small instances with no more than 40 customers are also reported at the end.

3.4.1 Parameter sensitivity

We have studied the impact of the number of consecutive iterations at each level and percentage of destruction (*% dstr.*) on the algorithmic performance. To this end, we have used a test set made of the RC2 instances with 100, 400 and 800 customers. The results are shown in Table 3.1. We have considered 100, 200 and 400 consecutive iterations at each level (i.e., 300, 600 and 1200 iterations for the whole workday-route-customer level sequence). Three different intervals for the percentage of destruction of the current solution were also tested, namely [5%, 35%], [35%, 65%] and [65%, 95%]. When a removal operator is chosen, a percentage value is uniformly randomly selected in the corresponding interval and applied at the appropriate level. For each possible combination of the two parameters and for each problem size, Table 3.1 shows the average percentage of unserved customers (*% unsv.*), total distance (*dist.*) and computation time in seconds (*CPU*).

Not surprisingly, the computation time increases with the percentage of des-

% dstr.	size	100 iterations % unsv.	100 iterations dist.	100 iterations CPU (s)	200 iterations % unsv.	200 iterations dist.	200 iterations CPU (s)	400 iterations % unsv.	400 iterations dist.	400 iterations CPU (s)
	100	28.1	1938.0	42.8	27.8	1923.5	42.4	28.1	1924.9	44.7
05-35	400	28.0	10940.6	671.6	28.1	10938.0	678.5	28.3	10938.2	671.6
	800	30.2	22713.3	2822.9	30.6	22698.9	2898.5	30.3	22830.2	2867.9
	100	31.4	1950.9	44.6	32.2	1949.2	44.8	31.5	1960.7	46.8
35-65	400	29.9	11157.0	745.0	30.0	11185.9	740.3	29.8	11258.1	755.9
	800	30.9	22915.8	2966.8	30.9	22976.3	2990.6	31.0	22850.0	2938.1
	100	34.0	1966.7	48.5	33.7	1963.7	45.7	35.6	1989.3	52.3
65-95	400	30.3	11277.1	823.6	30.0	11199.6	807.9	30.1	11230.7	806.7
	800	30.5	22896.3	3228.2	30.4	22774.4	3285.2	30.4	22677.3	3224.2

TAB. 3.1 – Impact of % of destruction and number of iterations at each level

truction because it is more costly to reinsert a larger number of customers into the current solution. Furthermore, solution quality tends to degrade. Based on the results shown in Table 3.1, the best combination is a percentage of destruction in the interval [5%, 35%] and 200 consecutive iterations at each level (i.e., 600 iterations for the whole workday-route-customer sequence and 40 such sequences over 24,000 iterations). This setting has been used in the following sections.

3.4.2 Results

Based on the best parameter setting identified in section 3.4.1, different variants of our ALNS have been applied to the whole set of test instances. These results are reported in Table 3.2. As indicated in this table, five different variants have been considered : *Cb* only uses the customer-based removal operators, *Cb/Rb* uses both the customer- and route-based removal operators while *Cb/Rb/W* uses all removal operators. Two variants of *Cb/Rb/W* have also been tested : *Cb/Rb1/W*, where the related route removal operator using the *MinD*

proximity measure is discarded (thus, only the random route removal and the related route removal using the *GC* measure are considered) and *Cb/Rb2/W*, where the related route removal using the GC measure is discarded (thus, only the random route removal and the related route removal using the *MinD* measure are considered). In each entry, we show the percentage of unserved customers (%), the total distance and the computation time in seconds (s), in this order. These results are averaged over all sizes for each problem class in Table 3.3.

The first observation is that the exploitation of a multi-level scheme is very beneficial when compared to the classical customer-based approach. By introducing a route level, the percentage of unserved customers is reduced by 7.81%, 8.22% and 6.92% on problem classes *RC*, *R* and *C*, respectively. An additional improvement, although less important, is also obtained by introducing the workday level.

The differences observed between *Cb/Rb1/W* and *Cb/Rb2/W* also indicate that the related route removal operator using the *MinD* proximity measure is superior to the one using the gravity center-based measure. Also, by comparing *Cb/Rb2/W* and *Cb/Rb/W*, a single related route removal operator based on the *MinD* measure provides better results than the use of the two operators concurrently.

3.4.3 Comparison with optimal solutions

The best ALNS variant *Cb/Rb2/W* has been applied to small instances for which the optimum is known and reported in [5]. These instances have been created by considering only subsets of 25 and 40 customers in Solomon's original instances [68]. The reported optima have been obtained with $t_{max} = 75$ and a fleet of 2 vehicles. Table 3.4 reports average results obtained over all instances of

given class and size. On the 25-customer instances, ALNS is quasi-optimal. All customers are served and the gap in total distance does not exceed 1%. On the 40-customer instances of type RC and C, ALNS is also close to the optimum. On the three instances of type R, a difference of 2.5% is observed with regard to the percentage of unserved customers and a gap of 16% with regard to the total distance. But, these average gaps are largely due to only one particular instance, for which 3 customers are left unserved by ALNS (no unserved customer in the optimal solution) and a gap close to 30% is observed in total distance.

size	class	Cb	Cb/Rb	Cb/Rb/W	Cb/Rb1/W	Cb/Rb2/W
100	RC	27.0%	25.5%	24.8%	25.6%	25.7%
		1909.6	1892.9	1894.2	1900.4	1899.2
		27.8s	29.0s	27.9s	27.5s	28.1s
	R	13.5%	11.5%	10.9%	12.2%	10.9%
		1866.1	1828.8	1828.1	1838.8	1828.6
		29.9s	35.5s	32.7s	34.3s	33.5s
	C	0.0%	0.0%	0.0%	0.0%	0.0%
		2393.0	2239.6	2269.3	2221.3	2232.9
		41.2s	36.0s	42.8s	46.7s	43.2s
200	RC	20.3%	14.6%	12.3%	12.8%	12.4%
		8690.8	9202.2	9205.9	9262.9	9218.3
		130.1s	148.5s	135.3s	143.3s	140.3s
	R	13.5%	7.4%	6.4%	6.4%	6.2%
		10360.2	10577.7	11126.6	11239.8	11103.7
		125.6s	131.9s	129.4s	157.3s	126.9s
	C	3.1%	0.0%	0.0%	0.0%	0.0%
		9994.1	9750.4	9818.9	9806.7	9730.3
		103.0s	110.0s	138.1s	125.0s	126.2s
400	RC	33.8%	25.6%	23.2%	23.2%	22.4%
		9633.5	10310.4	10279.3	10311.8	10128.0
		500.0s	522.8s	475.1s	488.9s	460.4s
	R	18.4%	8.5%	6.3%	6.5%	6.2%
		11784.2	12102.1	12766.0	12903.3	12657.7
		511.9s	526.5s	438.4s	496.5s	427.7s
	C	7.6%	0.2%	0.1%	0.1%	0.0%
		11425.1	11949.6	11256.8	11921.0	10937.2
		471.4s	466.0s	349.3s	385.7s	340.0s
600	RC	32.7%	21.8%	21.2%	21.1%	20.4%
		14170.0	14384.5	15860.2	15831.1	15577.9
		1170.8s	1211.5s	1127.4s	1099.3s	1114.2s
	R	20.4%	10.4%	6.4%	6.5%	6.4%
		17023.7	18903.3	19270.8	19400.4	19089.2
		1281.5s	1291.4s	1025.0s	1139.1s	1009.1s
	C	23.3%	13.0%	10.1%	10.5%	9.9%
		14187.8	14225.4	14667.9	14803.7	14626.0
		1184.2s	1360.4s	1046.9s	1129.8s	1028.5s
800	RC	36.5%	26.1%	25.0%	25.2%	24.3%
		18353.3	18878.8	20772.6	20921.4	20858.5
		1938.4s	1993.2s	1818.2s	1955.7s	1842.8s
	R	22.2%	11.3%	8.0%	8.3%	7.9%
		22270.1	23890.9	26192.8	26402.5	26136.9
		2088.4s	2125.2s	1691.2s	1834.0s	1678.3s
	C	37.9%	26.9%	25.0%	24.8%	24.9%
		14333.8	14271.9	14427.7	14454.6	14441.1
		1861.7s	2071.3s	1857.1s	1810.6s	1747.4s
1000	RC	37.5%	30.3%	27.3%	27.5%	26.3%
		22063.0	25157.9	25635.0	25584.2	25368.3
		3008.5s	3331.5s	3050.5s	2899.5s	2855.2s
	R	24.7%	14.2%	10.6%	10.8%	10.3%
		26862.1	28876.2	30700.5	30845.8	30732.2
		3377.0s	3358.1s	2730.8s	2943.5s	2718.8s
	C	49.6%	39.8%	36.9%	36.9%	36.5%
		14994.9	14670.6	14700.3	14637.2	14587.6
		2508.9s	2804.8s	2819.6s	2807.1s	2602.4s

TAB. 3.2 – Average results by problem classes and sizes

class	Cb	Cb/Rb	Cb/Rb/W	Cb/Rb1/W	Cb/Rb2/W
RC	31.8%	24.0%	22.3%	22.6%	21.9%
	12470.0	13304.5	13941.2	13968.6	13841.7
	1129.3s	1206.1s	1105.7s	1102.4s	1073.5s
R	18.8%	10.6%	8.1%	8.5%	8.0%
	15027.7	16029.8	16980.8	17115.4	16924.7
	1235.7s	1244.7s	1007.9s	1100.8s	999.0s
C	20.2%	13.3%	12.0%	12.0%	11.9%
	11221.5	11184.6	11190.2	11307.4	11092.5
	1028.4s	1141.4s	1042.3s	1050.8s	981.3s

Tab. 3.3 – Average results over all sizes for each problem class

			Cb/Rb2/W		Optimal	
size	class	# instances	% unsv.	distance	% unsv.	distance
25	RC	5	0.0	844.1	0.0	844.0
	R	11	0.0	624.1	0.0	620.1
	C	7	0.0	635.6	0.0	629.1
40	RC	3	13.9	1386.4	13.3	1377.5
	R	3	3.3	1289.0	0.8	1071.6
	C	3	0.0	1105.0	0.0	1093.0

Tab. 3.4 – Comparison with optimal solutions

3.5 Conclusion

This paper has described an adaptation of the ALNS framework based on the hierarchical structure of the vehicle routing problem with multiple trips. Empirical results demonstrate that it is very beneficial to apply operators at the customer, route and workday levels, as opposed to the classical approach where only customer-based operators are used.

Chapitre 4

A Dynamic Vehicle Routing Problem with Multiple Delivery Routes

Résumé. Cet article s'intéresse à un problème de tournées de véhicules où chaque véhicule réalise des tournées multiples au cours de sa journée de travail et oú de nouvelles requêtes apparaissent de façon dynamique. La méthodologie proposée repose sur une recherche à voisinage variable développée précédemment pour le cas statique. Dans le cas dynamique, plusieurs scénarios sont considérés touchant à l'occurrence des requêtes futures afin de décider de l'opportunité d'accepter ou de refuser la requête courante. Des résultats expérimentaux comparent cette approche avec une approche myope où une nouvelle requête est acceptée dès qu'elle est réalisable.

A Dynamic Vehicle Routing Problem with Multiple Delivery Routes

Nabila Azi
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Michel Gendreau
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Jean-Yves Potvin
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT),
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Case postale 6128, succursale Centre-ville,
Montréal H3C 3J7, Canada.

Cet article a été soumis pour publication dans *Annals of Operations Research*

Abstract. This paper considers a vehicle routing problem where each vehicle performs delivery operations over multiple routes during its workday and where new customer requests occur dynamically. The proposed methodology for addressing the problem is based on an adaptive large neighborhood search heuristic, previously developed for the static version of the problem. In the dynamic case, multiple possible scenarios for the occurrence of future requests are considered to decide about the opportunity to include a new request into the current solution. It is worth noting that the real-time decision is about the acceptance of the new request, not about its service which can only take place in some future routes (a delivery route being closed as soon as a vehicle departs from the depot). In the computational results, a comparison is provided with a myopic approach which does not consider scenarios of future requests.

Keywords : Dynamic vehicle routing, multiple routes, scenarios, acceptance rule, adaptive large neighborhood search.

4.1 Introduction

A relatively recent development in the field of vehicle routing relates to the study of dynamic variants, where information about the problem is revealed as the current routes are executed by the vehicles. In general, the dynamic aspect comes from the occurrence of new customer requests, although a few papers address other types of events, like dynamic travel times (see, for example, [25, 54]).

In this paper, a previous algorithm developed for a vehicle routing problem with multiple delivery routes is applied in a dynamic setting where customer requests occur dynamically and must be responded to in real-time. It is worth noting that the most stringent real-time decision is about accepting or not a new request, not about finding the best possible way to integrate it into the current solution, since the request can only be included in some future route (a delivery route being closed as soon as the corresponding vehicle departs from the depot). This is illustrated in Figure 4.1 for a single vehicle's workday. When the new customer request is received, the vehicle is currently moving between customers i and j in route 1. Given that the vehicle has already departed from the depot (black square) to execute route 1, this route is closed. Only planned routes 2 and 3, which contain customers that have been assigned to the vehicle but will only be served after the return of the vehicle to the depot, can accept the new request. This problem is inspired from e-grocery applications where perishable goods are delivered to customers, thus leading to multiple short vehicle routes where the last customer in each route must be served within a given time limit from the route start time.

The rule for accepting a new request is based on multiple possible scenarios for the occurrence in time and space of future requests. Given that a mix of true and expected requests are found in the solution associated with each scenario, the

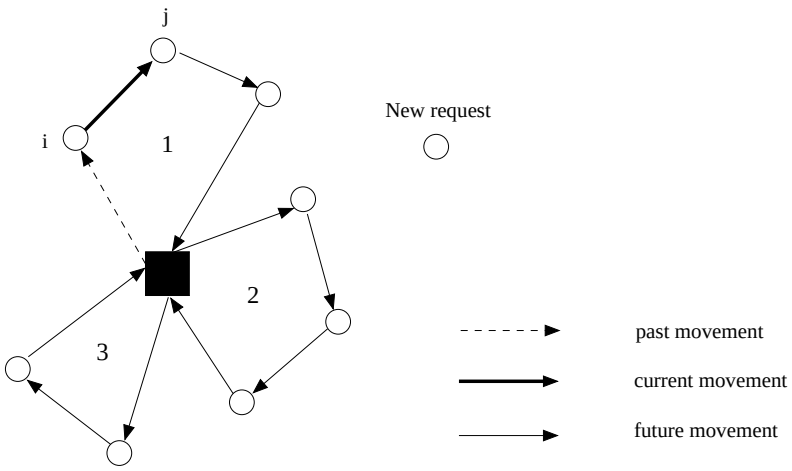

FIG. 4.1 – Current vehicle's workday

adverse effects of a myopic decision rule are alleviated. In the literature, future requests are either considered in an implicit way (see, for example, the double horizon approach [47]), or explicitly using different approaches for exploiting information about requests to come, including relocation and waiting strategies [8, 12, 35, 36, 48]. However, these methods are used to guide the integration of new requests into the current solution, not to accept or reject new requests based on some profit measure.

The remainder of the paper is as follow. In Section 4.2 the problem is defined. Section 4.3 then briefly describes the algorithm previously developed for the static version of the problem. The dynamic environment, as well as the proposed methodology for deriving a non myopic acceptance rule, are found in Section 4.4. Finally, computational results in Section 4.5 report a comparison between the non myopic and myopic decision rules (i.e., with and without scenarios).

4.2 Problem definition

The static version of the problem is defined on a directed graph $G = (V, A)$ with $V = \{0, 1, 2, ..., n\}$ the vertex set and A the arc set. Vertex 0 is the depot while the remaining vertices are customers. With each customer $i \in V \setminus \{0\}$ is associated a gain or revenue g_i, a service or dwell time s_i and a time window $[a_i, b_i]$, where a_i and b_i are the earliest and latest time, respectively, to start the service (with $a_0 = 0$ and $b_0 = \infty$). Thus, a vehicle has to wait if it arrives at customer i before a_i. With each arc $(i, j) \in A$ is associated a distance d_{ij} and a travel time t_{ij}. We also have a set $K = \{1, 2, ..., m\}$ of vehicles to deliver goods from the depot to customers. The duration of each route is limited by forcing the last customer to be served within t_{max} time units of the route start time. This restriction leads to short routes that must be combined and sequenced to form vehicle workdays. Also, a setup time for loading the vehicle, noted σ_r, is associated with each route r in the solution. This setup time is proportional to the sum of service times over all customers in the route. The objective is to maximize the total profit, which is the total gain associated with served customers minus the total traveled distance. A mathematical description of this problem can be found in [5].

In the dynamic version of the problem, the customers are not known in advance but must be responded to as the routes for serving previously assigned customers are executed by the vehicles. Once a customer request is received, an answer must be provided in real-time about the ability of the fleet to accommodate or not the request. On the other hand, the real-time aspect is much less stringent in the case of the actual service to the customer. As a pure delivery problem is dealt with, a route is fixed as soon as the corresponding vehicle departs from the depot to serve it. Accordingly, a new customer request can only be inserted in routes that will be executed later during a vehicle's workday.

4.3 Problem-solving methodology

An Adaptive Large Neighborhood Search (ALNS) [63] has been previously developed in [4] for solving the static version of the problem. First, an initial solution is constructed with an insertion heuristic. Then, a local search heuristic based on a large neighborhood is applied to improve this solution. This is explained in the following

4.3.1 Insertion heuristic

An insertion heuristic is used to construct an initial solution or to insert new customer requests in the current solution (in the dynamic setting). Every customer is inserted at its best feasible insertion place over every route in every workday, including an empty workday if one is still available. In this work, the best insertion place corresponds to the smallest detour in distance, where the detour is $d_{ji} + d_{il} - d_{jl}$ for the insertion of customer i between customers j and l. If there is no feasible insertion place for customer i, then each route is considered in turn and split into two subroutes, with an additional copy of the depot between the two subroutes. Once the original route is split, the insertion of the customer can take place in any of the two new routes. Each route is split in every possible way (i.e., at every customer location along the route) to find the best insertion place. If there is still no feasible insertion place, then customer i is left aside.

4.3.2 Large neighborhood search

For improving the solution, a large neighborhood structure is obtained through the use of solution destruction and reconstruction operators. These operators exploit the hierarchical nature of the problem by working either at the customer,

route (made of customers) or workday (made of routes) level. The adaptive feature comes from a weight associated with each operator. This weight is modified depending if the corresponding operator is successful or not in finding improved solutions. Clearly, an operator is more likely to be selected and applied to the current solution if its corresponding weight is larger.

A generic description of this problem-solving methodology is shown in pseudo-code in Algorithm 1, where $s*$ is the best known solution and where the acceptance criterion is probabilistic and based on simulated annealing ideas. For more details, the reader is referred to [4].

Procedure 4.3.1 ALNS

1. construct a feasible solution s ;

2. $s* \leftarrow s$;

3. initialize weights ;

4. while the stopping criterion is not met **do**

4.1 for L = workday, route, customer **do**

4.1.1 for I iterations **do**

a. probabilistically select a destruction operator at level L and a reconstruction operator based on their current weights ;

b. apply the destruction and reconstruction operators to s to obtain s' ;

c. if s' satisfies the acceptance criterion **then**

$s \leftarrow s'$;

if s' is better than $s*$ **then** $s* \leftarrow s'$;

4.1.2 adjust weights ;

5. return $s*$.

4.4 Dynamic environment

The algorithm developed for the static version of the problem was integrated into the new dynamic environment. This is described in the following, starting with the acceptance rule for new requests.

4.4.1 Acceptance rule

We have a set S of s-solutions, where each s-solution is based on a possible scenario for the occurrence of future requests. These s-solutions are used to evaluate the profitability of any new incoming request. The scenarios are based on some probabilistic knowledge that could be obtained, for example, through historical data. It is worth noting that the use of scenarios is also reported in [8]. However, only one scenario is associated with a solution and this scenario is used to guide the insertion of new requests into the current solution, not to decide about their acceptance. Acceptance rules where metrics based on true and expected requests are developed can be found in [16]. The authors assume that all potential customer locations are known in advance and a service request probability is associated with each location.

Here, it is assumed that the customer requests are received according to independent time-space Poisson processes. Basically, the service area is a grid made of Z squared zones and the horizon is divided into T time periods. The request arrival intensity within each zone z at time period t is λ_{zt} with $\sum_{z=1}^{Z} \lambda_{zt} = \lambda_t$, $t = 1, 2, ..., T$. Within each zone, the request is located uniformly randomly. Finally, the gain follows a normal law with average $4 \times \max_{i \in V \setminus \{0\}} d_{0i}$ and standard deviation $2 \times \max_{i \in V \setminus \{0\}} d_{0i}$, where d_{0i} is the distance between the depot and customer i. Given that the normal law is left-truncated at one standard de-

viation below the average, it is always profitable to serve a customer, since the gain exceeds the additional traveled distance needed to serve the customer (note that non profitable customers would always be rejected).

At the start, each s-solution contains only the expected requests in a given scenario and is optimized with our ALNS (see Section 4.3). Then, as time unfolds, true requests come in and are inserted in the s-solutions. Hence, a mix of true and expected requests will be found in these s-solutions. In fact, each s-solution is a blueprint for evaluating the opportunity value of new incoming requests. When a new request cannot be incorporated into the true solution (made only of true requests), due to the time constraints, it is automatically rejected. Otherwise, the opportunity value of the new request is calculated as the sum of the differences over all s-solutions of the s-solution quality with and without the insertion of the new request. If δ_i^s denotes this difference for request i and s-solution s, then the opportunity value of i is $\sum_{s \in S} \delta_i^s$. If this value is positive then the new request is accepted, otherwise it is rejected. When accepted, the new request is finally inserted in the true solution and in every s-solution. This chain of events is summarized in the pseudo-code below.

> If there is no feasible insertion place for the new request i in the true solution ts then reject i
>
> else
>
> 1. consider the insertion of the new request i in every s-solution ;
> 2. calculate $\Delta = \sum_{s \in S} \delta_i^s$;
> 3. if $\Delta > 0$ then accept i and insert it in ts and in every s-solution
>
>   else reject i.

Since the true requests are incorporated into each s-solution with the insertion heuristic described in Section 4.3.1, these s-solutions are reoptimized with ALNS

after the occurrence of I new requests, to maintain s-solutions of sufficiently good quality (in our computational results, $I = 10$).

4.4.2 Dynamic environment

There are two types of events that ask for a reaction : the occurrence of a new request and the departure of a vehicle from the depot. This is described in the following pseudo-code.

if "event" then

1. remove obsolete portion of every solution ;
2. if event is "occurrence of a new request" then
   - 2.1 apply the acceptance rule ;
   - 2.2 apply ALNS to the true solution ts ;
3. if event is "vehicle departure" then
   - 3.1 fix the vehicle's current route in ts ;
   - 3.2 update every s-solution ;
   - 3.3 apply ALNS to the true solution ts.

It is worth noting that ALNS is run on the planned portion of solution ts in steps 2.2 and 3.3 (excluding the current route, which is fixed). Removing the obsolete part of every solution in step 1 consists in removing customer requests that have been served since the occurrence of the previous event. In the case of a vehicle departure, the update in step 3.2 is aimed at maintaining the consistency of every s-solution with the true solution ts. That is, the current route should be the same, as well as the set of true requests to be served in the planned routes (although they are not necessarily served in the same order). Consider the example

in Figure 4.2 where the sequences of customers in the true solution ts and in some s-solution are illustrated (for simplicity, expected customers in the s-solution are not shown). When the current route of the s-solution needs to be replaced by the current route of the true solution ts (due to vehicle departure), customer 2 now appears twice, while customer 3 is missing. To maintain consistency, the following repair actions are taken :

1. When a customer appears twice, the second occurrence of this customer is simply removed from the planned routes.
2. When a customer is missing, the customer is reintroduced into the planned routes using the insertion heuristic of Section 4.3.1.

After the repair, the s-solution is said to be consistent with the true solution.

4.5 Computational results

A simulator was developed to test the proposed acceptance rule in different operating scenarios and compare it to a myopic approach where each new request is accepted as long as it is feasible. In the following, the characteristics of the simulator are first described. Then, a comparison is provided between the myopic and non myopic decision rules with an increasing number of requests over the same fixed horizon. Finally, the impact of the number of scenarios on solution quality is presented.

4.5.1 Simulator

The service area is a 5km $\times$ 5km square divided into into a grid of $Z = 2 \times 2 = 4$ zones. Within that square, the depot is located at coordinates $(2.0\text{km}, 2.5\text{km})$.

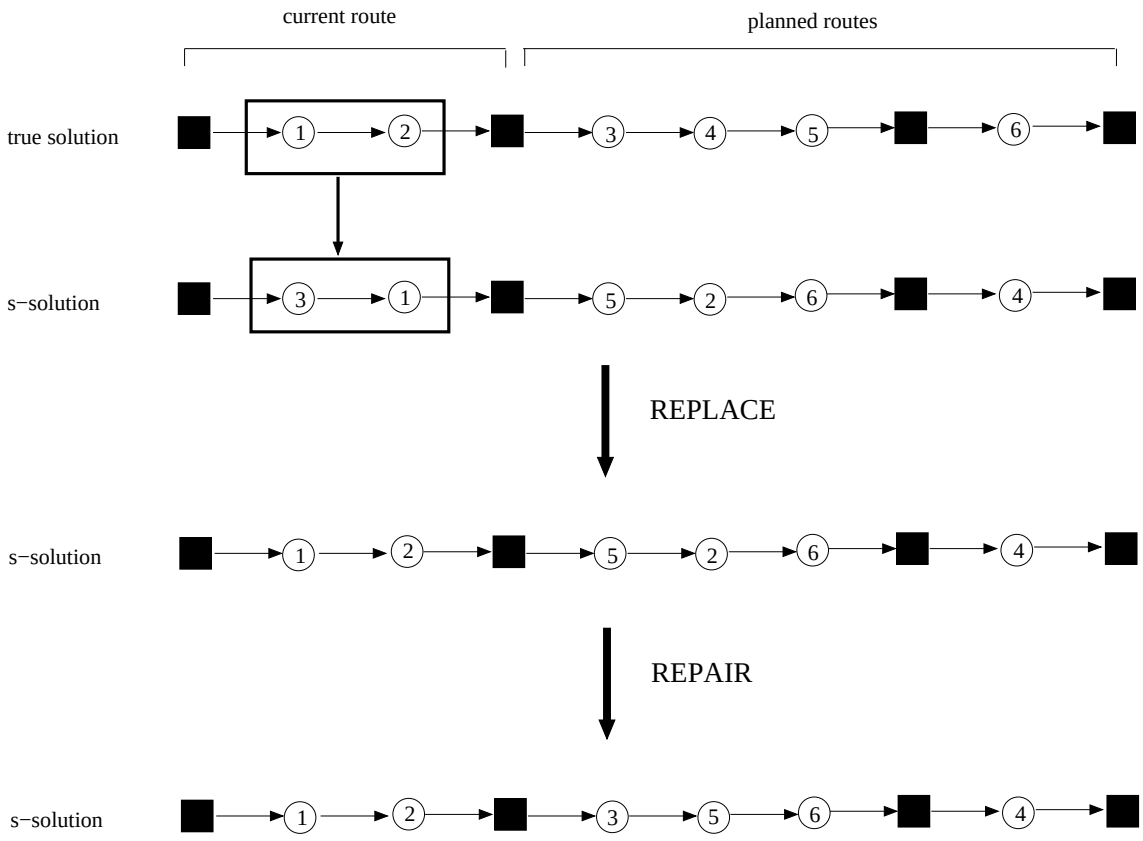

FIG. 4.2 – Maintaining the consistency of every s-solution

The time horizon is divided into $T = 4$ periods of one hour each. All vehicles move at a constant average speed of 30 km per hour. The service or dwell time is equal to 5 minutes at each customer location and the t_{max} value is set to 40 minutes. As previously mentioned, the new requests are received according to independent time-space Poisson processes where λ_{zt} is the request arrival intensity in zone z at time period t. Within each zone, the request is located uniformly randomly. In the computational results, the intensity is the same in each zone and in each time period, for the first three time periods. In the last period, no request is generated because it is too late to serve them the same day. The time windows of the new customer requests correspond to a full time period of one hour and are

equally distributed among the time periods that follow the current one (which is the time period associated with the occurrence of the new request). Finally, the gain follows a normal law with average $4 \times \max_{i \in V \setminus \{0\}} d_{0i}$ and standard deviation $2 \times \max_{i \in V \setminus \{0\}} d_{0i}$, where d_{0i} is the distance between the depot and customer i.

4.5.2 Comparison between myopic and non myopic approaches

In this section, we compare the performance of the myopic and non myopic acceptance rules with an increasing number of customer requests. This increase is obtained by varying the intensity parameter in the Poisson law to obtain 24, 36 and 48 customer requests per hour, on average (for a total of 72, 108 and 144 customers, respectively). Tables 4.1 and 4.2 show the results for the myopic approach and the non myopic approach, respectively, with 80 scenarios and using fleets of 3 and 5 vehicles. Each number is an average taken over 10 different instances.

As we can see, the non myopic approach provides a significant improvement over the myopic one. For a fleet of 3 vehicles, the profit increase in percentage varies between 7.5% and 16.8%. With 5 vehicles, the improvement is somewhat smaller but still stands between 4.8% and 9.8%. In addition to the profit increase, the solutions obtained with the non myopic approach systematically visit a larger number of customers. For both methods, an increase in the number of served customers and profit is observed when the number of vehicles increases from 3 to 5, for the same number of customers. Each vehicle also performs a smaller number of routes during its workday.

Through additional experiments, we also observed that the gap between the two approaches vanishes as the average number of customers per vehicle per hour

Number of vehicles	Number of customers per hour	Number of served customers	% of served customers	Number of routes per workday	Number of customers per route	Profit	CPU (s)
3	72.2	44.5	61.8	2.8	5.5	447.9	0.6
	108.4	55.4	51.2	3.3	6.0	564.7	1.7
	144.2	56.1	39.0	3.3	5.8	579.0	2.7
5	72.2	51.6	71.8	1.7	6.3	534.3	1.2
	108.4	69.4	64.1	2.3	6.2	710.8	3.8
	144.2	82.3	57.2	3.0	5.6	855.3	7.6

TAB. 4.1 – Simulation of 4 hours with the myopic approach, fleets of 3 and 5 vehicles and an increasing number of customers

Number of vehicles	Number of customers	Number of served customers	% of served customers	Number of routes per workday	Number of customers per route	Profit	CPU (s)
3	72.2	48.6	67.8	2.1	8.1	508.9	204.7
	108.4	57.5	53.1	2.1	9.2	606.9	272.8
	144.2	62.6	43.6	2.4	8.6	676.3	471.1
5	72.2	55.6	77.3	1.5	7.8	587.0	305.6
	108.4	70.2	64.9	1.6	8.9	745.0	733.4
	144.2	83.1	57.9	1.8	9.2	901.6	963.7

TAB. 4.2 – Simulation of 4 hours with the non myopic approach, fleets of 3 and 5 vehicles and an increasing number of customers

becomes very high (30 customers per hour per vehicle) or very low (2 customers per hour per vehicle). In the former case, the overwhelming complexity of the problem does not allow any method to take the edge. In the latter case, the simplicity of the problem has a similar impact. Between these two values, the non myopic approach is systematically better than the myopic one, although the gap can vary a lot, even for the same vehicle load. For example, by considering instances with 24 requests per hour and 2 vehicles, 36 requests per hour and 3 vehicles and 48 requests per hour and 4 vehicles, for the same average load of 12 customers per vehicle per hour, profit increases of 17.0%, 9.4% and 6.0%, respectively, have been obtained.

Finally, Table 4.3 illustrates the impact of the number of scenarios on the

performance of the non myopic approach, based on simulations with 3 vehicles and 36 requests per hour on average. An improvement is observed up to about 80 scenarios (with a corresponding increase in computation times). Beyond that point, the performance of the method reaches a plateau.

Number of scenarios	Number of customers	Number of served customers	% of served customers	Number of routes per workday	Number of customers per route	Profit	CPU (s)
10	108.4	55.8	51.5	2.1	9.0	587.0	36.7
20	108.4	57.0	52.7	2.2	8.9	596.7	75.6
40	108.4	57.2	52.9	2.0	9.4	600.0	137.8
80	108.4	57.5	53.1	2.1	9.2	606.9	272.8
120	108.4	57.7	53.3	2.2	9.0	607.9	433.3
150	108.4	57.6	53.2	2.1	9.2	607.0	500.4

TAB. 4.3 – Simulation of 4 hours with 3 vehicles, 36 requests per hour on average and an increasing number of scenarios

4.6 Conclusion

This paper shows the benefits of accounting for future customer requests when deciding about the acceptance or rejection of a new request in a dynamic setting where each vehicle executes multiple routes during its workday. Each decision is based on the planned routes to be executed later (thus, excluding the current route) due to the delivery nature of the problem. The use of multiple possible scenarios for the occurrence of new requests is also shown to be beneficial by providing solutions with an increased profit.

Chapitre 5

Conclusion

Cette thèse s'est consacrée à l'étude d'un problème de gestion de flottes de véhicules où un même véhicule est autorisé à effectuer plusieurs tournées au cours de sa journée de travail. Deux objectifs concurrents ont été considérés, soit la minimisation des coûts de transport et la maximisation des gains lors de la visite aux clients. En pratique, ce type de problème est particulièrement pertinent dans le cadre de la livraison de denrées périssables. Dans ce qui suit, nous présentons un résumé des principales contributions de cette thèse et nous suggérons de nouvelles avenues de recherche à explorer.

5.1 Principales contributions

Le chapitre 2 présente un algorithme exact basé sur une méthode de génération de colonnes où le sous-problème est un problème de plus court chemin élémentaire avec contraintes de ressources. C'est, à notre connaissance, la première méthode exacte qui ait été développée pour ce type de problème. L'algorithme est basé sur une méthode de génération de colonnes où le sous-problème correspond à

un plus court chemin élémentaire avec contraintes de ressources. Afin d'accélérer l'algorithme, chaque sous-problème est d'abord résolu de façon heuristique en relaxant la règle de dominance entre les chemins générés. La résolution exacte du sous-problème prend effet seulement lorsque l'heuristique n'est plus en mesure d'identifier de chemins avantageux. De plus, une méthode de stabilisation, déjà rapportée dans la littérature, a été utilisée afin d'améliorer les performances de l'algorithme. Des résultats expérimentaux sont rapportés pour des instances allant jusqu'à 40 clients.

Dans le chapitre 3, une approche heuristique permettant de résoudre des problèmes de plus grande taille exploite un voisinage de type "détruire et recréer". Plusieurs opérateurs de retrait et d'insertion de clients ont été développés et adaptés au problème qui présente une structure hiérarchique à trois niveaux, soit les journées de travail, les routes et les clients. On retrouve donc des opérateurs différents selon les niveaux, ce qui constitue une innovation par rapport à l'approche classique qui n'opère qu'au niveau client. Un mécanisme adaptatif ajuste les poids des opérateurs agissant aux différents niveaux afin de choisir les opérateurs à appliquer à chaque itération, au sein d'une structure globale de recherche qui alterne entre les différents niveaux de manière cyclique. De plus, lors de l'insertion d'un client, une route dans la journée de travail d'un véhicule peut être divisée en deux sous-routes, s'il n'y a aucune autre alternative réalisable, afin d'intégrer le plus de clients possibles dans la solution. Des résultats numériques avec des instances allant jusqu'à 1000 clients sont rapportés. Les bénéfices de l'approche hiérarchique par rapport à l'approche classique, qui agit seulement au niveau des clients, sont démontrés.

Dans le chapitre 4, nous nous sommes attaqués à une version dynamique du problème où les requêtes des clients sont reçues en temps réel. La stratégie d'acceptation des requêtes se fonde sur une connaissance probabiliste de la demande future afin d'améliorer la prise de décision au temps courant. Plus précisément,

des scénarios possibles quant à l'arrivée des requêtes futures sont générés et exploités afin de décider si une nouvelle requête doit être acceptée ou non. Les bénéfices de cette approche par rapport à une approche myope, où les requêtes futures ne sont aucunement tenus en compte, sont ensuite démontrés.

5.2 Perspectives de recherche

Plusieurs avenues de recherche intéressantes restent à explorer. En voici quelques exemples :

- Nous croyons qu'il serait possible d'améliorer significativement les temps d'exécution de la méthode exacte. Compte tenu de la complexité du sous-problème de plus court chemin élémentaire avec contraintes de ressources, une approche heuristique pourrait être envisagée afin d'identifier des chemins de coût réduit négatif avant de recourir à la méthode exacte.
- L'approche de résolution heuristique pourrait être enrichie par l'introduction de nouveaux opérateurs de construction et de destruction. De plus, de nouvelles approches heuristiques faisant appel à d'autres concepts que ceux de la recherche adaptative à voisinage large, pourraient être développées afin de mieux situer les performances de cette dernière.
- Dans le cas du problème dynamique, on pourrait imaginer la présence de fenêtres multiples pour le service à chacun des clients. La stratégie d'acceptation ou de refus pourrait alors intégrer une tarification variable selon les fenêtres de temps encore disponibles à un client. Ceci permettrait une réduction des coûts d'opération puisque la compagnie de livraison pourrait favoriser les fenêtres de temps qui entrainent les coûts les moins élevés, tout en facturant davantage les clients qui seraient prêts à payer plus cher afin d'être servis dans la fenêtre de temps qu'ils considèrent optimale.

- Certaines idées développées dans cette thèse, telle la stratégie d'acceptation ou de refus des nouvelles requêtes, pourraient être appliquées dans d'autres contextes comme le transport par camion en charge pleine ("truckload trucking").
- Il serait intéressant d'aborder d'autres problématiques rencontrées en pratique dans un contexte dynamique, par exemple le bris d'un véhicule qui force une révision en temps réel de la solution courante.

Bibliographie

[1] F. Alonso, M.J. Alvarez, and J.E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59 :963–976, 2008.

[2] C. Archetti, A. Hertz, and M.G. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13 :49–76, 2007.

[3] N. Azi, M. Gendreau, and J-Y. Potvin. An exact algorithm for a single vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178 :755–766, 2007.

[4] N. Azi, M. Gendreau, and J.-Y. Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple trips. Technical report, CIRRELT-2010-08 Montréal, submitted to *Computers & Operations Research*, 2010.

[5] N. Azi, M. Gendreau, and J-Y. Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202 :756–763, 2010.

[6] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 46 :316–329, 1998.

[7] M. Battara, M. Monaci, and D. Vigo. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 13 :316–329, 2008.

[8] R. Bent and P. Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52 :977–987, 2004.

[9] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5 :211–230, 2007.

[10] J. Brandão and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100 :180–191, 1997.

[11] J. Brandão and A. Mercer. The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, 49 :799–805, 1998.

[12] J. Branke, M. Middendorf, G. Noeth, and M. Dessouky. Waiting strategies for dynamic vehicle routing. *Transportation Science*, 39 :298–312, 2005.

[13] S.E. Butt and T.M. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21 :101–111, 1994.

[14] S.E. Butt and D.M. Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26 :427–441, 1999.

[15] A.M. Campbell and M. Savelsbergh. Efficient insertion heuristics for vehicle routing ans scheduling problems. *Transportation Science*, 38 :369–378, 2004.

[16] A.M. Campbell and M. Savelsbergh. Decision support for consumer direct grocery initiatives. *Transportation Science*, 39 :313–327, 2005.

[17] I.-M. Chao, B.L. Golden, and E. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88 :475–489, 1996.

[18] I.-M. Chao, B.L. Golden, and E. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88 :464–474, 1996.

[19] G. Desaulniers, J. Desrosiers, and M. Solomon. *Column Generation*. Springer, 2005.

[20] J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis. Time constrained routing and scheduling. In *Network Routing*, pages 35–139. Elsevier Science B.V., 1995.

[21] D. Feillet. Problèmes de tournées avec gains : étude et application au transport inter-usines. *Thèse de Doctorat, Laboratoire Productique Logistique, École Centrale de Paris*, 2001.

[22] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39 :188–205, 2005.

[23] M. Fishetti, M. Salazar González, and P.. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10 :133–148, 1998.

[24] B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Technical report, Fachbereich Wirtschaftswissenschaften Universität, Hamburg, Germany, 1990.

[25] B. Fleischmann, S. Gnutzmann, and E. Sandvoss. Dynamic vehicle routing based on online traffic information. *Transportation Science*, 38 :420–433, 2004.

[26] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EU-ROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science*, volume 2, pages 57–64, 1999.

[27] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Seguin. Neighborhood search heuristics for a dynamic vehicle disparching problem with pick-ups and deliveries. *Transportation Research Part C*, 14 :157–174, 2006.

[28] M. Gendreau, F Guertin, J.-Y. Potvin, and E. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33 :381–390, 1999.

[29] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective travelling salesman problem. *Networks*, 32 :263–273, 1998.

[30] M. Gendreau, G. Laporte, and F. Semet. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106 :539–545, 1998.

[31] B.L. Golden, A. Assad, and R. Dahl. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems*, 7 :181–190, 1984.

[32] B.L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34 :307–318, 1987.

[33] B.L. Golden, Q. Wang, and L. Liu. A multifaceted heuristic for the orienteering problem. *Naval Research Logistics*, 35 :359–366, 1988.

[34] C. Gueguen. Méthodes de résolution exacte pour les problèmes de tournées de véhicules. *Thèse de Doctorat, Laboratoire Productique Logistique, École Centrale de Paris*, 1999.

[35] L.M. Hvattum, A. Lokketang, and G. Laporte. Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40 :421–438, 2006.

[36] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40 :211–225, 2006.

[37] V. Kamaraïnen, J. Saranen, and J. Holmström. The reception box impact on home delivery efficiency in the e-grocery business. *International Journal of Physical Distribution and Logistics Management*, 31 :414–426, 2001.

[38] M. Kantor and M.B. Rosenwein. The team orienteering problem. *European Journal of Operational Research*, 43 :629–635, 1992.

[39] C.P. Keller. Algorithms to solve the orienteering problem : A comparaison. *European Journal of Operational Research*, 41 :224–231, 1989.

[40] H.K. Kilpala. The impact of electronic commerce on transport and logistics in the retail grocery industry. Technical report, University of Oulu, Finland, 1999.

[41] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1990.

[42] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26 :193–207, 1990.

[43] A. Larsen, O. Madsen, and M. Solomon. Partially dynamic vehicle routing - models and algorithms. *Journal of the Operational Research Society*, 53 :637–646, 2002.

[44] Y.-C. Liang and E. Smith. An ant colony approach to the orienteering problem. *Journal of the Chinese Institute of Industrial Engineers*, 23 :403–414, 2006.

[45] I.L. Lin and H. Mahmassani. Can online grocers deliver ? some logistics considerations. *Transportation Research Record*, 1817 :17–24, 2002.

[46] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Computing Journal*, 44 :2245–2269, 1965.

[47] M. Mitrović, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38 :669–685, 2004.

[48] M. Mitrović and G. Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38 :635–655, 2004.

[49] R. Montemanni and L. Gambardella. Ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences*, 34 :287–306, 2009.

[50] A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34 :28–47, 2007.

[51] R.J. Petch and S. Salhi. A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133 :69–92, 2003.

[52] R.J. Petch and S. Salhi. A genetic algorithm based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, 6 :591–613, 2007.

[53] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problem. *Computers & Operations Research*, 34 :2403–2435, 2007.

[54] J.-Y. Potvin, X. Ying, and I. Benyahia. Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 33 :1129–1137, 2006.

[55] M. Punakivi and M.J. Holmström. Extending e-commerce to the grocery business - the need for reengineering fleet management in distribution. *Logistik Management*, 2, 2000.

[56] M. Punakivi and M.J. Holmström. Identifying the success factors in e-grocery home delivery. *International Journal of Retail and Distribution Management*, 29 :156–163, 2001.

[57] M. Punakivi, H. Yrjola, and M.J. Holmstrom. Solving the last mile issue : Reception box or delivery box ? *International Journal of Physical Distribution and Logistics Management*, 31 :427–439, 2001.

[58] R. Ramesh and K.M. Brown. An efficient four-phase heuristic for the generalized orienteering tour problem. *Computers & Operations Research*, 18 :151–165, 1991.

[59] R. Ramesh, Y. Yoon, and M. Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4 :155–165, 1992.

[60] G. Righini and M. Salani. Bounded bidirectional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3 :255–273, 2006.

[61] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with dynamic programming. *Computers & Operations Research*, 4 :1191–1203, 2009.

[62] Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicule routing. *Journal of Heuristics*, 1 :147–167, 1995.

[63] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time window. *Transportation Science*, 40 :455–472, 2006.

[64] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35 :660–668, 2007.

[65] H. Schneider and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159 :139–171, 2000.

[66] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, 1520 :417–431, 1998.

[67] J. Smarös, M.J. Holmström, and V. Kamaraïnen. New service opportunities in the e-grocery business. *International Journal of Logistics Management*, 11 :61–74, 2000.

[68] M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35 :254–265, 1987.

[69] W. Souffriau, P. Vansteenwegenb, G..V. Berghea, and D.V. Oudheusdenb. A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37 :1853–1859, 2010.

[70] E. Taillard, G. Laporte, and M. Gendreau. Vehicle routing with multiple use of vehicules. *Journal of the Operational Research Society*, 47 :1065–1070, 1996.

[71] H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32 :1379–1407, 2005.

[72] F. Tasgetiren. A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research*, 4 :1–26, 2002.

[73] P. Toth and D. Vigo. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, 2002.

[74] L. Tsiligirides. Heuristics methods applied to orienteering. *Journal of the Operational Research Society*, 35 :797–809, 1984.

[75] P. Vansteenwegen, W. Souffriau, G.V. Berghe, and D.V. Oudheusden. Iterared local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36 :3281–3290, 2009.

[76] Q. Wang, X. Sun, B.L. Golden, and J. Jiyou. Using artificial neural networks to solve the orienteering problem. *Annals Of Operations Research*, 61 :111–120, 1995.

[77] J. Yang, P. Jaillet, and H. Mahmassani. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38 :135–148, 2004.

[78] H. Yrjola. Physical distribution considerations for electronic grocery shopping. *International Journal of Physical Distribution and Logistics Management*, 31 :746–761, 2001.

[79] Q.H. Zhao, S.Y. Wang, K. Lai, and G. Xia. A vehicle routing problem with multiple use of vehicles. *Advanced Modeling and Optimization*, 4 :21–40, 2002.