Université de Montréal

Résolution d'un problème de collecte et livraison dynamique sur un réseau routier avec temps de parcours variables

par

Félix Caron

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique, option Recherche opérationnelle

Mars 2022

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Résolution d'un problème de collecte et livraison dynamique sur un réseau routier avec temps de parcours variables

présenté par

Félix Caron

a été évalué par un jury composé des personnes suivantes :

Nadia El-Mabrouk

(président-rapporteur)

Jean-Yves Potvin

(directeur de recherche)

Michel Gendreau

(codirecteur)

Emma Frejinger

(membre du jury)

Résumé

Les services de livraison express font face au défi d'optimiser les routes de leurs véhicules alors que ceux-ci circulent dans un réseau routier où les temps de parcours varient en fonction du moment de la journée et où ils doivent répondre à l'arrivée dynamique de requêtes consistant à récupérer et livrer des colis. Notre but ici est de proposer une modélisation et une méthode de type heuristique pour résoudre ce problème. Nous commençons par explorer les travaux menés précédemment au sujet de l'arrivée dynamique des requêtes, des temps de parcours variables selon le moment de la journée et des collectes et livraisons dans les problèmes de tournées de véhicules. Ensuite, nous décrivons le problème de manière formelle sur le graphe du réseau routier avec des requêtes deux-points où l'objectif est de minimiser le temps total de parcours des véhicules et les temps de retard aux points de service et au dépôt. Par la suite, nous détaillons l'implémentation d'une méthode de résolution basée sur la recherche tabou utilisant une structure de voisinage basée sur la réinsertion d'une requête. Cette méthode utilise également la structure *Dominant Shortest Path* (DSP) qui considère plusieurs chemins alternatifs entre chaque paire de sommets, contrairement à l'approche traditionnelle où un chemin unique est fixé a priori. Finalement, nous testons notre méthode à l'aide de 390 instances générées de manière synthétique afin d'évaluer son efficacité ainsi que l'impact de certains aspects du problème et de la méthode de résolution. Les résultats démontrent une amélioration particulièrement importante due à l'utilisation de la structure DSP.

Mots-clés : Recherche opérationnelle, tournées de véhicules, recherche tabou, collecte et livraison, dynamique, réseau routier, temps de parcours variables, *Dominant Shortest Path*

Abstract

Express delivery services face the challenge of optimizing the routes of their vehicles while they are moving in a road network where the travel times vary according to the time of day in order to serve dynamic requests which consist in collecting and delivering parcels. Our goal here is to propose a model and a heuristic method to solve this problem. We begin by exploring previous work on the topic of the dynamic arrival of requests, time-dependent travel times and pickups and deliveries in vehicle routing problems. Afterwards, we describe the problem formally on the graph of the road network with the objective of minimizing the total travel time of the vehicles and lateness at the service points and at the depot. Then, we detail the implementation of a solving method based on tabu search using a neighbourhood structure based on the reinsertion of a request. This method also uses the Dominant Shortest Path (DSP) structure which considers multiple alternative paths between each pair of vertices, unlike the traditional approach where a single path is fixed a priori. Finally, we test our method using 390 instances generated synthetically in order to evaluate its efficiency as well as the impact of certain aspects of the problem and solution method. The results show a particularly significant improvement due to the use of the DSP structure.

Keywords : Operations research, vehicle routing, tabu search, pickup and delivery, dynamic, road network, time-dependent, dominant shortest path

Table des matières

Liste des tableaux

Liste des figures

Liste des algorithmes

Liste des sigles et des abréviations

DSP	*Dominant Shortest Path*, structure contenant divers chemins alternatifs entre les paires de sommets et l'évolution de leur temps de parcours tout au long de la journée.
FIFO	*First-in-first-out*, propriété spécifiant qu'un véhicule commençant à parcourir un arc avant un autre véhicule arrivera toujours à destination avant ce dernier.

Remerciements

Je souhaite remercier mon directeur de recherche Jean-Yves Potvin et codirecteur Michel Gendreau pour avoir accepté de m'accompagner tout au long de ce projet. Toujours prêts à partager leurs vastes connaissances et expérience dans le domaine avec moi, ils ont été d'une aide inestimable à sa réalisation.

Merci également au Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT) de m'avoir accueilli et soutenu financièrement.

Chapitre 1

Introduction

Au cours des dernières décennies, une quantité importante de travaux en recherche opérationnelle a porté sur les problèmes de transport. Cela n'est pas un hasard puisque d'importants coûts y sont associés dans un grand nombre d'industries et qu'il peut être difficile d'optimiser ces coûts.

Par exemple, pour les problèmes de tournées de véhicules qui servent à modéliser les situations où l'on dispose d'une flotte de véhicules afin de visiter un ensemble d'emplacements, de nombreux algorithmes ont été élaborés afin de déterminer quel véhicule doit visiter quels emplacements et dans quel ordre afin de minimiser les coûts engendrés.

Alors que la recherche à leur sujet progresse, on ajoute de nouveaux aspects à leur modélisation afin de mieux représenter des situations réelles spécifiques et afin de s'adapter à de nouvelles situations.

Le problème auquel nous nous intéresserons ici correspond à un service de livraison express qui reçoit des requêtes de clients consistant à récupérer un colis à un certain emplacement et à le livrer à un autre emplacement au cours de la même journée. Étant donné les avancées technologiques au niveau de la communication entre les clients, la centrale de répartition et les véhicules, ces requêtes deux-points peuvent être reçues et intégrées aux routes de manière dynamique, c'est-à-dire au cours de la journée alors que les véhicules sont déjà en déplacement. Nous ne disposons cependant d'aucune information probabiliste à l'avance quant à leur arrivée. De plus, afin d'avoir une modélisation qui est fidèle à la réalité des villes où ce genre de service est offert, nous utiliserons un réseau routier composé d'intersections et de segments d'axes routiers sur lesquels le temps de parcours change de façon connue en fonction du moment de la journée.

Bien que chacun des aspects de notre problème aient déjà été abordés dans la littérature, qu'il s'agisse de requêtes deux-points, du traitement dynamique de ces requêtes ou encore des temps de parcours qui varient en fonction du moment de la journée, la contribution de ce mémoire vient de leur considération simultanée dans un même problème. Des méthodes connues ont donc été adaptées afin de résoudre ce problème d'une grande complexité.

Nous proposons une approche de résolution où, à l'arrivée de chaque nouvelle requête, un problème statique d'optimisation des routes courantes des véhicules est défini avec la nouvelle requête et les requêtes déjà affectées aux véhicules mais non encore desservies. Une méthode de recherche tabou est utilisée pour réaliser cette optimisation jusqu'à l'arrivée de la prochaine requête. Cette méthode a été choisie puisqu'elle a souvent été utilisée pour résoudre des problèmes de tournées de véhicules autant statiques que dynamiques et que, malgré sa simplicité, elle produit généralement de bons résultats en peu de temps de calcul.

Nous présenterons d'abord au Chapitre 2 les travaux qui ont été menés sur des problèmes similaires. Nous définirons ensuite notre problème au Chapitre 3 et la méthode de résolution au Chapitre 4. Enfin, nous analyserons au Chapitre 5 certains résultats obtenus pour des instances générées de manière synthétique.

Chapitre 2

Revue de littérature

Dans ce chapitre, nous décrivons brièvement des travaux de recherche qui ont déjà été effectués au sujet de problèmes ayant des aspects en commun avec le nôtre. Ceux-ci sont divisés en trois grandes catégories, soit les problèmes dynamiques, les problèmes avec temps de parcours variables et les problèmes de collecte et livraison.

2.1. Dynamisme

Un aspect important de notre problème de tournées de véhicules est le dynamisme. Un article intéressant pour se familiariser avec ce concept est la revue de littérature de Pillac et al. [**19**]. On y explique d'abord la pertinence de ce genre de problèmes pour répondre à des besoins du monde réel. On y précise également la différence entre les problèmes dynamiques qui, contrairement aux problèmes statiques, considèrent l'évolution de l'information disponible au cours de leur résolution, et les problèmes stochastiques qui, contrairement aux problèmes déterministes, considèrent l'incertitude de l'information, souvent par l'utilisation de distributions de probabilités. Dans notre cas, il s'agira d'un problème dynamique et déterministe. Plusieurs applications de ces types de problèmes sont ensuite résumées et quelques méthodes de résolution générales sont présentées.

Pour les problèmes dynamiques et déterministes, une approche fréquemment utilisée est la ré-optimisation continue. Un article de Gendreau et al. [**9**] est donné comme exemple. Dans cet article, les auteurs présentent d'abord une méthode de recherche tabou avec mémoire adaptative pour la version statique du problème de tournées de véhicules avec fenêtres de temps qui tire avantage d'une exécution parallèle. Le but de la mémoire adaptative est de conserver à tout moment les meilleures solutions rencontrées et de les exploiter pour créer de nouvelles solutions de départ pour la recherche tabou. La méthode de résolution développée

pour le problème statique est ensuite adaptée pour la version dynamique en identifiant deux types d'événements : l'arrivée d'une nouvelle requête et le départ d'un véhicule vers sa prochaine destination. Entre chacun de ces événements, la procédure de recherche tabou est exécutée afin d'optimiser le problème statique défini à l'aide des requêtes courantes. Chaque fois qu'un nouvel événement survient, la recherche tabou est interrompue et la meilleure solution trouvée est ajoutée à la mémoire adaptative, si cette dernière n'est pas encore remplie ou si la solution est meilleure que la pire solution en mémoire. Après le traitement approprié des solutions en mémoire adaptative en fonction de l'événement (par exemple, dans le cas de l'arrivée d'une nouvelle requête, celle-ci est insérée dans chacune des solutions), une solution de départ est construite à partir des solutions en mémoire adaptative pour optimisation par la recherche tabou, et ce, jusqu'à l'arrivée du prochain événement. Cette approche a ensuite été adaptée à des variantes du problème en permettant la diversion [**12**], en utilisant des temps de parcours variables dans un graphe de clients [**13**], en exploitant des connaissances probabilistes [**14**] et en desservant des requêtes deux-points avec collecte et livraison [**8**].

Psaraftis et al. [**20**] recensent un grand nombre de méthodes, autres que la recherche tabou, qui ont été utilisées pour résoudre des problèmes de tournées de véhicules dynamiques. Dans la plupart des cas, il s'agit de méthodes ayant d'abord été utilisées pour résoudre la version statique du problème. Ces méthodes vont d'heuristiques simples faisant appel à la simple insertion des nouvelles requêtes [**3, 4**] jusqu'à des méthodes exactes de génération de colonnes [**5**], en passant par les algorithmes génétiques [**23**], les colonies de fourmis [**6**] et les essaims particulaires [**15**].

D'autres méthodes ont été conçues précisément pour les problèmes dynamiques et stochastiques en exploitant l'information probabiliste sur les demandes futures à l'aide de processus de décision markoviens [**24**] et de systèmes de files d'attente [**11**] par exemple.

2.2. Temps de parcours variables

Dans les villes, le trafic et donc le temps de parcours des segments routiers peuvent varier de façon importante mais prévisible selon le moment de la journée. Plusieurs travaux ont donc été menés dans le but d'intégrer ces variations connues d'avance aux problèmes de tournées de véhicules.

Afin d'obtenir des temps de parcours provenant d'un réseau routier réel, Ehmke et al. [**7**] font appel à des données historiques appelées FCD pour *Floating Car Data*. Ces données sont récupérées par des flottes de véhicules équipés d'appareils GPS comme les taxis par exemple. Grâce à l'évolution de leur position GPS au fil du temps, on peut connaître leur

vitesse à tout moment, ce qui est utilisé pour calculer la vitesse moyenne sur chaque axe routier à chaque heure de la semaine. Les axes où la variation de vitesse est similaire sont ensuite regroupés afin de simplifier le modèle et une fonction représentant la variation du temps de parcours est dérivée pour chaque groupe.

Attanasio et al. [**1**] utilisent quant à eux une approche différente qui consiste à estimer les temps de parcours à l'aide d'un réseau de neurones qui reçoit en entrée l'année, la semaine et la journée, ainsi que les conditions météorologiques, le trafic actuel et les zones de départ et d'arrivée déterminées à l'aide du code postal. Pour que le réseau de neurones soit suffisamment entraîné et produise de bonnes estimations, il est cependant nécessaire de disposer d'une importante quantité de données historiques.

Dans la suite, nous présentons différentes façons d'intégrer les variations dans les temps de parcours aux problèmes de tournées de véhicules.

2.2.1. Modélisation dans un graphe de clients

Dans plusieurs cas, on se base sur une modélisation classique avec un graphe complet où chaque sommet représente un client et chaque arc le chemin de longueur minimale entre deux clients dans le réseau routier sous-jacent. Ainsi, Malandraki et al. [**18**] ont défini un modèle où l'horizon de temps est divisé en intervalles et un temps de parcours différent est associé à chaque intervalle. Un problème avec cette approche est qu'il est parfois préférable d'attendre le prochain intervalle de temps avant de parcourir un arc afin d'arriver plus tôt à destination. Cela signifie qu'un véhicule commençant à parcourir un arc après un autre véhicule pourrait arriver à destination avant ce dernier, en attendant un peu afin de profiter d'un temps de parcours plus favorable pour parcourir l'arc. On dit qu'une telle situation ne vérifie pas la propriété FIFO, pour *first-in-first-out*. Bien que certaines modélisations du problème permettent les temps d'attente, il ne serait pas très réaliste ici de faire attendre un véhicule pour lui permettre d'arriver plus tôt à sa destination en empruntant le même chemin.

Ichoua et al. [**13**] obtiennent la propriété FIFO en associant une vitesse plutôt qu'un temps de parcours à chaque intervalle de temps. Une fonction linéaire par morceaux représentant l'évolution du temps de parcours peut ensuite être calculée pour chaque arc en considérant les variations de vitesse qui surviennent tout au long du parcours de l'arc.

2.2.2. Modélisation dans un multigraphe

En considérant que les temps de parcours des arcs d'un réseau routier (qui correspondent à des segments de rue entre deux intersections) changent au cours de la journée, il est tout à fait possible que le chemin le plus rapide entre deux clients change également au cours de la journée. La modélisation sous forme d'un graphe de clients ne permet évidemment pas d'en tenir compte puisqu'un seul chemin est considéré pour se rendre d'un client à un autre. Pour cette raison, Setak et al. [**22**] proposent plutôt une modélisation dans un multigraphe, c'est-à-dire un graphe où plus d'un arc peut relier les mêmes sommets. De cette façon, plusieurs chemins alternatifs peuvent être représentés et c'est l'algorithme d'optimisation qui décide du chemin emprunté.

2.2.3. Modélisation dans un réseau routier

Plus récemment, des auteurs ont exploité directement les informations disponibles dans le réseau routier, évitant ainsi de construire explicitement un multigraphe. Par exemple, Ben Ticha [**25**] résout un problème de tournées de véhicules avec fenêtres de temps à l'aide d'un algorithme *branch-and-price* opérant directement sur le graphe du réseau routier où les sommets représentent les intersections et les arcs représentent les segments d'axes routiers. Pour des fins de comparaison, deux graphes de clients sont également générés à partir du réseau. Pour le premier, chaque arc représente le chemin le plus court entre deux clients en fonction de la distance et, pour le deuxième, chaque arc représente l'ensemble des chemins étant le plus rapide à un moment de la journée. À tout moment, le coût d'un tel arc correspond au temps de parcours minimal parmi tous les chemins de l'ensemble. L'algorithme proposé pour obtenir ces ensembles a cependant une complexité en pire cas qui est exponentielle.

Gmira et al. [**10**] proposent également d'utiliser un graphe où chaque arc représente un ensemble de chemins possibles avec la structure DSP pour *Dominant Shortest Path*. Ces différents chemins sont obtenus à l'aide d'une variante de l'algorithme de Dijkstra avec temps de parcours variables en considérant des départs à différents moments de la journée. Des fonctions linéaires par morceaux représentant l'évolution du temps de parcours des différents chemins sont ensuite calculées et combinées avec une fonction minimum. Cette dernière est finalement utilisée lors de l'optimisation par recherche tabou pour connaître le temps de parcours entre deux sommets du réseau routier. Nous reviendrons sur cette approche à la Section 4.2.3.

2.3. Collecte et livraison

Battarra et al. [**2**] recensent un grand nombre de travaux effectués sur des problèmes où l'on doit à la fois effectuer des collectes et des livraisons. Ceux-ci sont divisés en trois grandes catégories : les problèmes *many-to-many* où un seul type d'item doit être récupéré à certains endroits et livré à d'autres, les problèmes *one-to-many-to-one* où un type d'item doit être récupéré à certains endroits et un autre type d'item doit être livré aux mêmes endroits ou des endroits différents, et les problèmes *one-to-one* où chaque item est unique et possède un lieu de collecte et un de livraison. Les méthodes de résolution pour ces problèmes sont très variées autant du côté des méthodes exactes que des heuristiques.

Les problèmes *one-to-one*, qui utilisent des requêtes deux-points, s'appliquent bien aux services de livraison rapide comme celui auquel nous nous intéressons. À cet effet, Gendreau et al. [**8**] ont adapté une méthode de recherche tabou en utilisant un voisinage basé sur des chaines d'éjection afin de générer de nouvelles solutions. L'insertion d'une requête dans une route se fait en deux temps. Le point de collecte est d'abord inséré au meilleur endroit possible selon une approximation et l'on insère ensuite le point de livraison de la même façon. Évidemment, les options sont moindres pour cette seconde insertion puisqu'elle peut seulement avoir lieu après le point de collecte.

Lu et al. [**17**] ont quant à eux opté pour une heuristique basée sur les insertions pour résoudre un problème avec requêtes deux-points et fenêtres de temps. Toutes les insertions possibles sont testées et évaluées en fonction de la distance additionnelle à parcourir, la réduction de la marge de temps entre l'arrivée du véhicule et la borne supérieure de la fenêtre de temps pour tous les clients sur la route, et de l'apparence de la route. Ce dernier élément est calculé à partir des croisements qui peuvent être observés lorsque la route est tracée en deux dimensions. Minimiser le nombre de véhicules utilisés faisant également partie de l'objectif, une nouvelle route n'est créée que lorsque l'insertion n'est réalisable dans aucune des routes actuelles. La performance de cette méthode est comparée avec celle d'une recherche tabou et, bien qu'elle soit inférieure, elle arrive à s'en rapprocher avec des temps de calcul beaucoup plus courts.

Une autre heuristique basée sur les insertions est proposée par Qu et al. [**21**]. Celle-ci détermine l'ordre d'insertion des requêtes en attribuant à chacune un score basé sur une mesure de regret qui représente en quelque sorte l'urgence de son insertion. En fait, le regret correspond à la différence en distance entre la solution résultant d'une insertion à la meilleure position possible et celles résultant d'autres insertions possibles. Les requêtes n'ayant qu'une seule bonne possibilité d'insertion sont ainsi identifiées et priorisées. Cette approche est comparée à une approche vorace classique qui choisit simplement la requête dont

l'insertion cause la plus petite augmentation de la distance. Dans les deux cas, un élément de hasard est utilisé afin de pouvoir appliquer ces heuristiques à répétition et obtenir différentes solutions afin de choisir la meilleure. Les résultats rapportés montrent que la performance des deux heuristiques est très semblable.

Chapitre 3

Description du problème

Dans ce chapitre, nous décrivons notre problème de manière formelle et détaillée, en commençant par le réseau routier, puis les véhicules, les requêtes à desservir et l'objectif à minimiser.

3.1. Réseau routier

Une façon assez commune de modéliser les problèmes de tournées de véhicules est à l'aide d'un graphe de clients. Dans un tel graphe, un sommet représente le point de départ des véhicules et les autres sommets représentent chacun un client. De plus, pour chaque paire de sommets, un arc les relie dans chaque direction avec un coût correspondant au temps de parcours du chemin le plus court dans le réseau routier sous-jacent. Cette modélisation a le bénéfice d'être simple, mais elle ne s'adapte malheureusement pas très bien à notre problème où l'emplacement des requêtes n'est pas toujours connu d'avance et où nous considérons des variations de vitesse sur les segments individuels du réseau routier.

Pour cette raison, nous représentons plutôt le réseau routier à l'aide d'un graphe orienté $G = (V, A)$ où les sommets V représentent les intersections et les arcs A représentent les segments d'axes routiers de notre réseau. Chaque arc $a \in A$ possède une distance d_a qui correspond à la longueur du segment et un profil de vitesse $s_a(t)$ qui prend la forme d'une fonction par paliers sur le domaine $[0, T]$ où T est la fin de la journée. La vitesse au temps T est conservée par la suite jusqu'à ce que tous les véhicules soient de retour au dépôt.

Afin de calculer le temps de parcours $w_a(t)$ d'un arc a pour un départ du sommet origine de l'arc au temps t, on doit considérer la vitesse au moment du départ ainsi que tous les changements de vitesse qui surviendront par la suite avant d'avoir parcouru la distance totale de l'arc. La Figure 1 présente un exemple simple de la façon dont le temps de parcours évolue

selon le profil des vitesses. Dans cet exemple, le premier intervalle correspond à un temps de départ entre 0 (inclusivement) et 3 minutes (exclusivement) et est associé à une vitesse de 30 km/h. Le second intervalle correspond à un temps de départ supérieur ou égal à 3 minutes et est associé à une vitesse de 60 km/h.

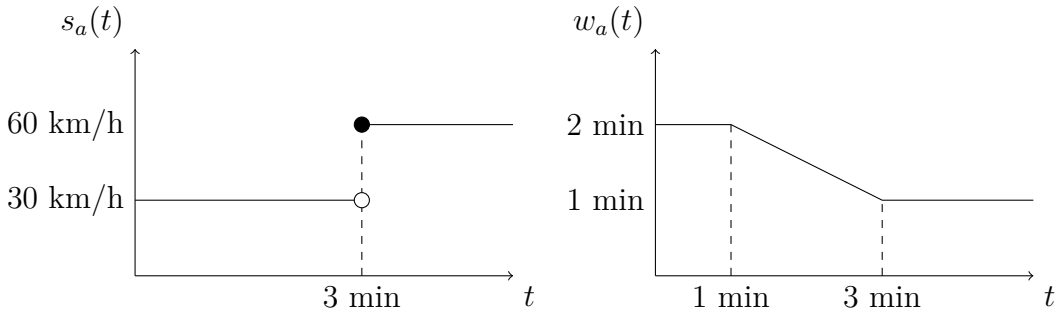

Figure 1. Profil de vitesse $s_a(t)$ et fonction du temps de parcours $w_a(t)$ correspondant pour un arc a d'une longueur $d_a = 1$ km.

La méthode pour calculer les temps de parcours à partir d'un profil de vitesse est décrite dans Ichoua et al. [**13**]. Cette modélisation est assez représentative du monde réel où les vitesses peuvent changer assez brusquement alors que les temps de parcours évoluent de manière plus graduelle. Elle permet également de respecter la propriété FIFO. En effet, un véhicule qui s'engage sur un arc avant un autre aura la chance de prendre une certaine avance et conservera cette avance jusqu'à l'arrivée à destination, puisque les deux véhicules progressent à la même vitesse à tout moment.

3.2. Véhicules

Pour servir les requêtes, nous disposons de m véhicules identiques. Chaque véhicule a une capacité Q et a accès aux mêmes arcs avec les mêmes temps de parcours.

Une solution à notre problème consistera donc à définir m routes commençant au sommet v_0 (représentant le dépôt central) au temps 0 et se terminant au même sommet v_0, idéalement avant la fin de la journée au temps T. Pour chaque véhicule dont le retour s'effectue plus tard, une pénalité sera ajoutée au coût de la solution.

Afin de garder la communication simple entre la centrale de répartition et les véhicules au cours de la journée, la diversion n'est pas permise. Autrement dit, lorsqu'un véhicule commence à se diriger vers sa prochaine destination, il n'est plus possible de rediriger le véhicule vers une autre destination.

3.3. Requêtes

Sur leur route, les véhicules devront desservir toutes les requêtes d'un certain ensemble R. Comme il s'agit d'un service de livraison dans la même journée, chaque requête $r \in R$ est constituée de deux points $r = (r^+, r^-)$ appelés points de service, où r^+ représente la collecte du colis au sommet $v_{r^+} \in V$ et r^- sa livraison au sommet $v_{r^-} \in V$.

Pour qu'une solution soit valide, la collecte et la livraison de chaque requête doivent être effectuées par le même véhicule et la collecte doit se trouver avant la livraison dans la route. D'autres points de service peuvent cependant se retrouver entre les deux.

On peut donc définir les ensembles suivants :

- L'ensemble des points de collecte : $R^+ = \{r^+ \mid (r^+, r^-) \in R\}$
- L'ensemble des points de livraison : $R^- = \{r^- \mid (r^+, r^-) \in R\}$
- L'ensemble des points de service : $R^\pm = R^+ \cup R^-$
- L'ensemble des sommets associés à un point de service : $V_{R^\pm} = \{v_{r^\pm} \mid r^\pm \in R^\pm\}$

Notons que $V_{R^\pm} \subseteq V$, c'est-à-dire que tous les sommets ne sont pas nécessairement associés à des points de service. Dans un tel cas, ces sommets ne représentent rien d'autre que de simples intersections dans le réseau routier.

En plus de garantir une livraison dans la même journée, il faut également offrir un service qui tient compte des disponibilités des clients au cours de cette journée. Pour ce faire, une fenêtre de temps $[b_{r^\pm}, e_{r^\pm}] \subseteq [0, T]$ est associée à chaque point de service $r^\pm \in R^\pm$. Ces fenêtres de temps sont souples. Cela signifie qu'un véhicule qui arrive au point de service $r^\pm$ avant la borne inférieure $b_{r^\pm}$ devra attendre jusqu'à ce temps avant d'y débuter le service alors qu'un véhicule qui arrive après la borne supérieure $e_{r^\pm}$ pourra débuter le service immédiatement, quoiqu'une pénalité pour son retard sera ajoutée au coût de la solution.

Comme nous nous intéressons à un problème dynamique, toutes les requêtes ne sont pas nécessairement connues au début de la journée. Chaque requête $r \in R$ a donc une heure d'arrivée $t_r \in [0, b_{r^+}]$. Avant cette heure d'arrivée, cette requête ne peut être incluse dans aucune solution puisqu'elle n'est pas encore connue. À partir de cette heure, et donc lorsque toutes les actions prévues avant cette heure ont été accomplies, les points de service de la requête doivent faire partie d'une solution pour que celle-ci soit valide pour le reste de la journée. Les requêtes $r \in R$ pour lesquelles $t_r > 0$ sont qualifiées de requêtes dynamiques, alors que les autres sont les requêtes statiques.

À chaque point de service $r^\pm \in R^\pm$ est également associé un temps de service $u_{r^\pm}$ qui représente le temps nécessaire pour effectuer la collecte ou la livraison. De plus, à chaque

requête $r \in R$ est associée une quantité q_r, appelée demande, qui représente la consommation de la capacité du véhicule par le colis. La capacité résiduelle du véhicule est donc réduite de cette quantité lorsque le point de collecte est visité et elle est augmentée de cette même quantité lorsque le point de livraison est visité. Ainsi, nous posons $q_{r^+} = -q_r$ et $q_{r^-} = q_r$ afin de tenir compte des changements dans la capacité résiduelle du véhicule après avoir desservi les points de service $(r^+, r^-) = r$. Pour qu'une solution soit valide, la capacité résiduelle ne peut être à aucun moment inférieure à 0.

3.4. Objectif

L'objectif de notre problème consiste à identifier des routes pour les véhicules qui, en plus de respecter les contraintes mentionnées plus haut, minimisent le temps total de parcours des véhicules plus les temps de retard aux points de service et au dépôt (à la fin de la journée).

Plus précisément, il s'agit de minimiser une somme pondérée de trois éléments :

$$\min \sum_{k=1}^{m} \sum_{(a,t)\in A_k} w_a(t) + \sum_{r^\pm \in R^\pm} \alpha x_{r^\pm} + \sum_{k=1}^{m} \beta y_k$$

où nous retrouvons les variables de décision :

- Les paires $(a, t) \in A \times [0, +\infty[$ dans l'ensemble A_k telles que le véhicule $k \in \{1, \ldots, m\}$ commence à parcourir l'arc a au temps t;
- Le temps de retard $x_{r^\pm}$ au point de service $r^\pm \in R^\pm$, c'est-à-dire le temps qui s'est écoulé entre la borne supérieure de la fenêtre de temps $e_{r^\pm}$ et le début du service à ce point. Lorsque le service débute avant la borne supérieure de la fenêtre de temps, $x_{r^\pm}$ est égal à 0;
- Le temps de retard y_k du véhicule $k \in \{1, \ldots, m\}$ au dépôt, c'est-à-dire le temps qui s'est écoulé entre T et le retour du véhicule au dépôt à la fin de la route. Lorsque le véhicule retourne au dépôt avant T, y_k est égal à 0;

et les constantes :

- α servant à établir l'importance des temps de retard aux points de service par rapport aux temps de parcours;
- β servant à établir l'importance des temps de retard au dépôt par rapport aux temps de parcours.

3.5. Illustration du problème

La Figure 2 présente un exemple simple de notre problème en cours de résolution. Initialement, on a deux requêtes. La requête A est assignée au véhicule 1 et la requête B est assignée au véhicule 2. Alors que le véhicule 1 a effectué la collecte A et se dirige vers le point de livraison correspondant, et que le véhicule 2 se dirige vers le point de collecte B, une nouvelle requête C survient. Ses points de service sont donc ajoutés à la route du véhicule 2 en modifiant les déplacements prévus de façon à minimiser le temps de parcours et les retards supplémentaires.

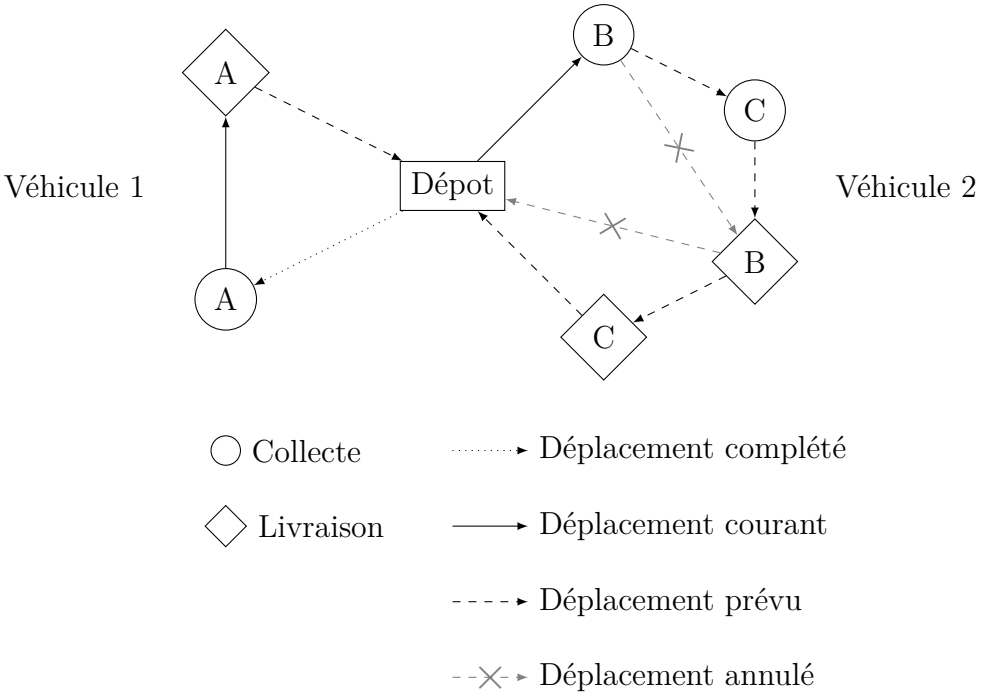

Figure 2. Illustration du problème

Chapitre 4

Méthode de résolution

Dans ce chapitre, nous présentons notre méthode de résolution basée sur la recherche tabou. Plus précisément, nous détaillons l'évaluation des temps de parcours avec la structure DSP, la gestion des temps d'attente, l'évaluation du coût des solutions, la méthode d'insertion des requêtes, la structure de voisinage, le critère tabou, le déroulement d'une itération et les phases de diversification.

4.1. Recherche tabou

La recherche tabou est une méthode fréquemment utilisée dans la littérature pour résoudre des problèmes de tournées de véhicules dynamiques semblables au nôtre, car elle permet généralement l'obtention de bons résultats en peu de temps. Nous utiliserons donc cette méthode en l'adaptant à notre réseau routier ainsi qu'aux requêtes deux-points.

La recherche tabou est une métaheuristique itérative qui, à chaque itération, considère la solution courante et cherche à l'améliorer. Pour ce faire, des modifications assez simples sont appliquées à la solution courante (par exemple, déplacement d'une requête, échange de deux requêtes qui se trouvent dans deux routes différentes, etc.) afin de produire un voisinage de nouvelles solutions. On sélectionne alors la meilleure solution dans ce voisinage, qu'elle soit meilleure ou non que la solution courante. On répète alors avec la solution sélectionnée qui devient la nouvelle solution courante. Afin d'éviter de cycler dans l'espace des solutions, on ajoute des interdictions qui empêchent de retourner à des solutions déjà visitées. Ces interdictions sont dites tabous, d'où le nom de la méthode.

Dans notre contexte dynamique, une solution de départ est d'abord construite avec les requêtes statiques, c'est-à-dire celles connues en début de journée. On applique ensuite la recherche tabou à cette solution jusqu'à l'arrivée d'une nouvelle requête. À ce moment,

cette nouvelle requête est d'abord insérée dans la meilleure solution produite par la recherche tabou. Puis, cette dernière est réappliquée à la solution qui inclut maintenant la nouvelle requête, et ce, jusqu'à l'arrivée de la prochaine requête. Il faut noter ici que la recherche tabou possède la propriété *anytime*, ce qui est très utile dans un contexte dynamique. Cette propriété signifie qu'une solution sera retournée, quel que soit le temps de calcul alloué à la recherche tabou, et que la qualité de la solution produite ne peut que demeurer la même ou s'améliorer lorsque le temps de calcul alloué augmente. De plus, puisque nous utilisons des fenêtres de temps souples et qu'il n'y a aucune limite de retard aux points de service et au dépôt, une solution réalisable existera toujours.

Notre implémentation de la recherche tabou sera présentée en détail à la section 4.6.

4.2. Temps de parcours

Lors de la recherche tabou, l'ordre de visite des points de service est susceptible de changer de nombreuses fois. Il est donc important de pouvoir mesurer rapidement le temps de parcours entre n'importe quelle paire de sommets $i, j \in V_{R^{\pm}} \cup \{v_0\}$ pour un départ à n'importe quelle heure de la journée. Puisqu'il peut exister un très grand nombre de chemins différents entre deux sommets et que le plus rapide n'est pas toujours le même au courant de la journée, nous allons considérer seulement un sous-ensemble P_{ij} de chemins possibles.

Ce sous-ensemble est constitué des chemins les plus rapides identifiés par une variante de l'algorithme de Dijkstra adaptée aux temps de parcours variables, où l'on considère des départs à différents moments de la journée, tel que décrit par Gmira et al. [**10**]. Ces moments sont ceux où les vitesses changent sur les arcs. Par exemple, dans une situation réelle, ils pourraient correspondre à l'heure de pointe du matin, au reste de l'avant-midi, à l'heure du midi, etc. L'algorithme est exécuté une fois pour chaque sommet de départ et moment de la journée, mais permet d'identifier le meilleur chemin vers tous les autres sommets à la fois.

Pour chaque chemin $p \in P_{ij}$, nous calculons ensuite une fonction $f_p(t)$ donnant le temps d'arrivée à j pour un départ de i au temps $t \in [0, +\infty]$. Puisque cette fonction est croissante, sa réciproque $f_p^{-1}(t)$ est aussi une fonction qui permet d'obtenir l'heure de départ correspondant à une heure d'arrivée.

Lorsqu'on voudra connaître le temps de parcours entre deux points de service aux sommets i et j pour un temps de départ t, les fonctions $f_p(t)$ associées aux différents chemins possibles $p \in P_{ij}$ pourront être comparées au temps t et le chemin ayant le temps de parcours le plus rapide sera choisi. Nous aurons donc une façon très efficace d'identifier le

meilleur chemin parmi les différents chemins possibles (il faut toutefois rappeler ici que nous ne considérons qu'un sous-ensemble de tous les chemins possibles).

4.2.1. Calcul de $f_p(t)$ pour les chemins constitués d'un seul arc

Pour le cas le plus simple où le chemin p est constitué d'un seul arc a, on utilise la fonction de vitesse $s_a(t)$ pour calculer $f_p(t) = f_a(t)$. Cette dernière sera une fonction linéaire par morceaux avec des points de rupture à chaque fois qu'une nouvelle vitesse commence ou cesse d'être utilisée au long du parcours. Pour chaque moment $t \in [0, T]$ où il y a un changement de vitesse dans $s_a(t)$, il y aura donc deux points de rupture $(t, f_a(t))$ et $(f_a^{-1}(t), t)$ dans $f_a(t)$.

Les valeurs de $f_a(t)$ et $f_a^{-1}(t)$ sont dans ce cas évaluées à l'aide des Algorithmes 1 et 2 où $t_1, \ldots, t_n$ sont les temps en ordre croissant où il y a des changements de vitesse sur l'arc a. La variable d est d'abord initialisée avec la longueur de l'arc (ligne 2) et, pour chaque palier de la fonction de vitesse $s_a(t)$ se terminant avant l'arrivée à destination, on retire la distance qui a pu être parcourue (ligne 5). Une fois arrivé au palier final, on ajoute le temps nécessaire afin de parcourir la distance restante (ligne 8). Pour $f_a(t)$, où l'on recherche le temps d'arrivée étant donné le temps de départ, les paliers sont parcourus en ordre croissant (ligne 7), alors que pour $f_a^{-1}(t)$, où l'on recherche le temps de départ étant donné le temps d'arrivée, les paliers sont parcourus en ordre décroissant (ligne 7).

Algorithme 1 : Calcul du temps d'arrivée $f_a(t)$

1 $t' \longleftarrow t$
2 $d \longleftarrow d_a$
3 $k \longleftarrow \max\{k' \mid k' \in \{1, \ldots, n\} \wedge t_{k'} \leq t\}$
4 **tant que** $k < n \wedge d > (t_{k+1} - t') \cdot s_a(t_k)$ **faire**
5 $\quad d \longleftarrow d - (t_{k+1} - t') \cdot s_a(t_k)$
6 $\quad t' \longleftarrow t_{k+1}$
7 $\quad k \longleftarrow k + 1$
8 **retourner** $t' + \frac{d}{s_a(t_k)}$

Lorsque $f_a^{-1}(t)$ donne un résultat négatif, le point de rupture $(f_a^{-1}(t), t)$ n'est pas conservé puisque cela correspondrait à un départ avant le temps 0, ce qui est impossible. Notons que l'ensemble $\{k' \in \{1, \ldots, n\} \mid t_{k'} \leq t\}$ ne sera jamais vide puisque $t_1 = 0$ et $t \in [0, T]$.

La Figure 3 présente un exemple simple de fonction $f_a(t)$ résultante. Notons que le dernier segment de cette fonction s'étend à l'infini avec une pente de 1 après le dernier changement de vitesse.

Algorithme 2 : Calcul du temps de départ $f_a^{-1}(t)$

$t' \longleftarrow t$
$d \longleftarrow d_a$
$k \longleftarrow \max\{k' \mid k' \in \{1, \ldots, n\} \wedge t_{k'} \leq t\}$
tant que $k > 1 \wedge d > (t' - t_k) \cdot s_a(t_k)$ **faire**
    $d \longleftarrow d - (t' - t_k) \cdot s_a(t_k)$
    $t' \longleftarrow t_k$
    $k \longleftarrow k - 1$
retourner $t' - \frac{d}{s_a(t_k)}$

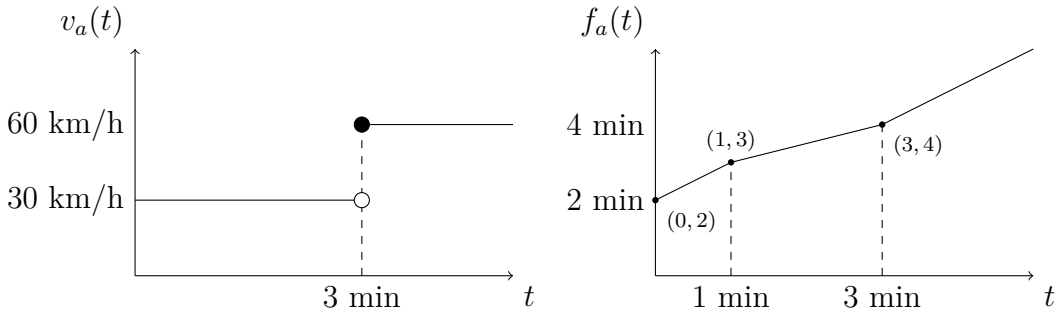

Figure 3. Profil de vitesse $v_a(t)$ et fonction du temps d'arrivée $f_a(t)$ correspondant pour un arc a d'une longueur $d_a = 1$ km.

4.2.2. Calcul de $f_p(t)$ pour les chemins constitués de plusieurs arcs

Pour les chemins constitués de plusieurs arcs, la fonction se définit de façon récursive : $f_p(t) = f_a(t) + f_{p'}(f_a(t))$ où a est le premier arc de p et p' est le reste du chemin. La fonction $f_p(t)$ aura alors les points de rupture $PR_{f_p} = \{(t_1, f_{p'}(t_2)) \mid (t_1, t_2) \in PR_{f_a}\} \cup \{(f_a^{-1}(t_1), t_2) \mid (t_1, t_2) \in PR_{f_{p'}}\}$ où PR_{f_a} et $PR_{f_{p'}}$ sont les ensembles des points de rupture de f_a et $f_{p'}$ respectivement. Encore une fois, les points de rupture $(f_a^{-1}(t_1), t_2)$ où $f_a^{-1}(t_1)$ donne un résultat négatif ne sont pas conservés.

La Figure 4 présente un exemple simple de fonction $f_p(t)$ résultante où les points $(0, 2)$, $(1, 3)$ et $(3, 4)$ de la fonction $f_a(t)$ sont remplacés par les points $(0, 3.5)$, $(1, 5.33)$ et $(3, 8)$ alors que les points $(2.5, 4)$ et $(4, 8)$ de la fonction $f_{p'}(t)$ sont remplacés par les points $(0.5, 4)$ et $(3, 8)$. Puisque $f_a^{-1}(0) < 0$, le point $(0, 1.5)$ n'est remplacé par aucun nouveau point.

4.2.3. Structure Dominant Shortest Path (DSP)

Afin de réduire le nombre de comparaisons à effectuer chaque fois qu'un temps de parcours doit être évalué, une option est de construire la fonction du temps d'arrivée minimum

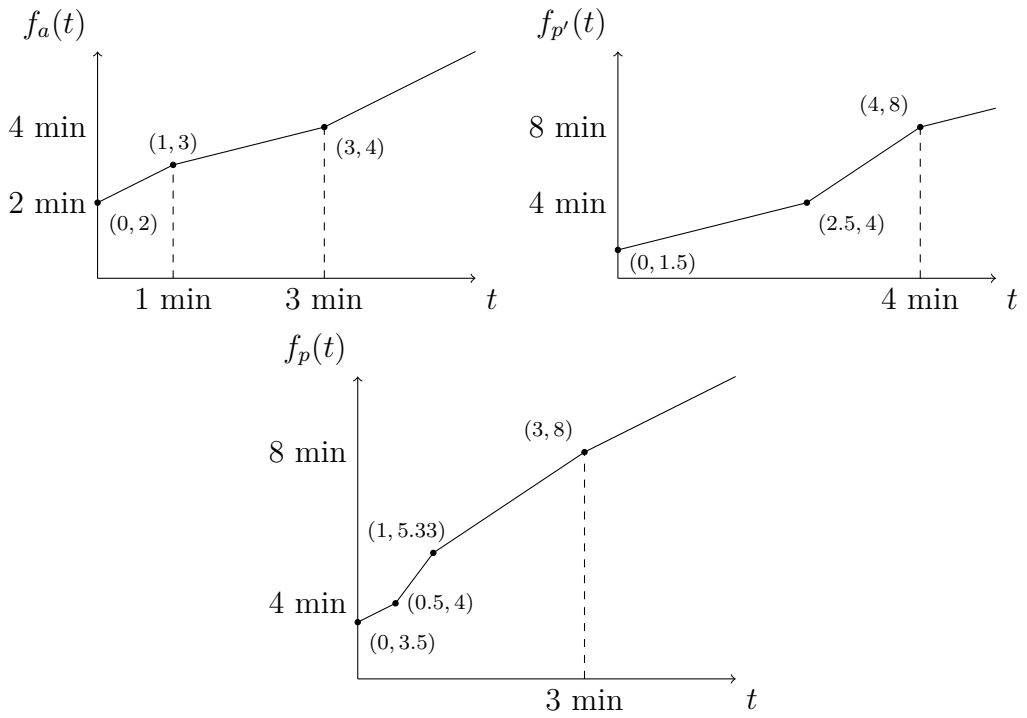

Figure 4. Composition de la fonction du temps d'arrivée $f_p(t)$.

$h_{ij}(t) = \min_{p \in P_{ij}} f_p(t)$. Cela peut être fait en prenant une section de la fonction associée au chemin le plus rapide au temps 0 jusqu'à la première intersection avec la fonction d'un autre chemin et de répéter ensuite avec ce nouveau chemin. Pour être en mesure d'indiquer plus tard au véhicule le chemin à emprunter, il faut aussi conserver en mémoire le chemin correspondant à chaque section de $h_{ij}(t)$.

Un exemple simple de cette méthode, qui correspond à la structure DSP proposée par Gmira et al. [**10**], est présenté à la Figure 5. Dans ce cas, trois chemins différents $P_{ij} = \{p_1, p_2, p_3\}$ ont été calculés pour se rendre du sommet i au sommet j. Les fonctions $f_{p_1}(t)$, $f_{p_2}(t)$ et $f_{p_3}(t)$ correspondantes sont chacune représentées par une couleur différente alors que la fonction $h_{ij}(t)$ est constituée de la ligne pleine. Pour la première section, du temps 0 à 0.8, le chemin p_1 (rouge) est le plus rapide et $h_{ij}(t) = f_{p_1}(t)$. Ensuite, de 0.8 à 2.67, le chemin p_2 (bleu) est le plus rapide et $h_{ij}(t) = f_{p_2}(t)$. Et finalement, à partir du temps 2.67, le chemin p_3 (vert) est le plus rapide et $h_{ij}(t) = f_{p_3}(t)$.

4.2.4. Consultation des fonctions $f_p(t)$ et $h_{ij}(t)$

L'Algorithme 3 détaille le processus de consultation des fonctions de temps d'arrivée $f_p(t)$ et $h_{ij}(t)$ où $(pr_{11}, pr_{12}) \ldots (pr_{n1}, pr_{n2})$ sont les n points de rupture en ordre croissant

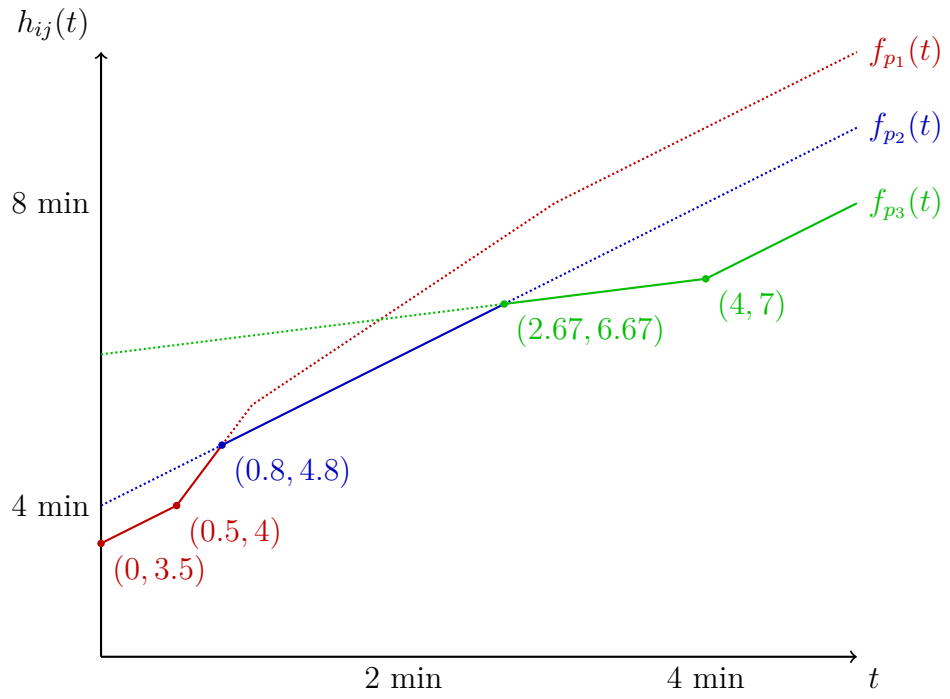

Figure 5. Fonction du temps d'arrivée minimum $h_{ij}(t)$.

composant la fonction. Notons que le premier élément de ces points correspond au temps de départ et que le deuxième correspond au temps d'arrivée.

Algorithme 3 : Consultation de $f_p(t)$ et $h_{ij}(t)$

1 **si** $t \geq pr_{n1}$ **alors**
2 $\quad$ **retourner** $t + pr_{n2} - pr_{n1}$
3 $l_1 \longleftarrow 1$
4 $l_2 \longleftarrow n$
5 **tant que** $l_1 < l_2 - 1$ **faire**
6 $\quad$ $l_3 \longleftarrow \left\lfloor \frac{l_1 + l_2}{2} \right\rfloor$
7 $\quad$ **si** $t < pr_{l_3 1}$ **alors**
8 $\quad\quad$ $l_2 \longleftarrow l_3$
9 $\quad$ **sinon**
10 $\quad\quad$ $l_1 \longleftarrow l_3$
11 **retourner** $pr_{l_1 2} + (t - pr_{l_1 1}) \frac{pr_{l_2 2} - pr_{l_1 2}}{pr_{l_2 1} - pr_{l_1 1}}$

Après le dernier point de rupture, le temps de parcours demeure toujours le même. Donc, si le temps de départ t est après pr_{n1}, on n'a qu'à ajouter à t ce temps de parcours, qui correspond à la différence entre pr_{n2} et pr_{n1} (ligne 2). Sinon, on doit identifier les deux points de rupture consécutifs entre lesquels se trouve le temps t en faisant appel à une

recherche dichotomique (lignes 5 à 10) et ensuite effectuer une interpolation linéaire entre les deux points de rupture (ligne 11).

La recherche dichotomique, qui réduit de moitié le nombre de points de rupture à considérer à chaque itération, est d'une complexité de l'ordre de $O(\log n)$ dans le pire des cas. Comme les autres opérations se font en temps constant, il s'agit également de la complexité pour la totalité de la consultation.

Dans le cas où l'on voudrait connaître le temps de départ pour un temps d'arrivée donné à l'aide des fonctions $f_p^{-1}(t)$ ou $h_{ij}^{-1}(t)$, le même algorithme est utilisé, mais les deux éléments de chaque point de rupture doivent être inversés.

4.3. Temps d'attente

Puisqu'une fenêtre de temps est associée à chaque point de service, il est possible de retrouver des temps d'attente dans les routes. Une attente se produit lorsqu'un véhicule est disponible au temps t pour quitter le sommet $i \in V_{R^\pm} \cup \{v_0\}$ afin de se rendre à son prochain point de service $r^\pm$ au sommet $j = v_{r^\pm}$ et que $h_{ij}(t) < b_{r^\pm}$, c'est-à-dire qu'un départ le plus tôt possible du sommet i cause une arrivée au sommet j avant la borne inférieure de sa fenêtre de temps $b_{r^\pm}$. Puisque le service ne peut commencer au sommet j avant $b_{r^\pm}$, nous avons le choix d'ajouter un temps d'attente avant le départ de i, après l'arrivée à j ou un mélange des deux.

Afin de minimiser le temps de parcours entre ces deux sommets, on pourrait choisir le temps de départ $t' \in [t, h^{-1}(b_{r^\pm})]$ qui minimise $(h(t') - t')$. Par contre, puisque nous sommes dans un contexte dynamique, nous préférons garder davantage de flexibilité en quittant i le plus tard possible, soit au temps $h^{-1}(b_{r^\pm})$. De cette façon, la prochaine destination j du véhicule pourra toujours être modifiée si une nouvelle requête arrive près de la localisation courante du véhicule ou encore si la recherche tabou identifie une meilleure solution entre temps. Notons que l'attente ne peut causer aucun retard sur le reste de la route puisqu'il est prévu que le véhicule arrive à la borne inférieure de la fenêtre de temps de sa prochaine destination j.

Dans le même esprit, on pourrait également faire attendre au sommet i un véhicule k ayant terminé sa route jusqu'au temps $h_{iv_0}^{-1}(T)$ avant de retourner au dépôt, toujours afin de garder un maximum de flexibilité dans un contexte dynamique. Cependant, nous croyons qu'il est plus réaliste de définir un temps $T' \in [\max_{r \in R} t_r, T]$ à partir duquel le véhicule est autorisé à retourner au dépôt. Dans une situation réelle, ce temps peut être vu comme une heure à partir de laquelle les nouvelles requêtes ne sont plus acceptées pour la journée

courante et seront plutôt desservies le lendemain. Puisqu'aucune nouvelle requête n'arrivera après T' et qu'il y a peu de chances qu'une requête provenant d'un autre véhicule soit transférée au véhicule k par la procédure d'optimisation, une attente plus longue ne serait pas justifiée. Le véhicule n'attendra donc jamais au sommet i après le temps $\min(T', h^{-1}_{iv_0}(T))$.

4.4. Évaluation du coût d'une solution

Pour évaluer le coût associé à une solution à partir de l'ordre de visite des points de service $r^{\pm}_{k1}, \ldots, r^{\pm}_{kn_k}$ pour chaque véhicule $k = 1 \ldots m$ où n_k est le nombre de points de service dans la route, on garde en mémoire pour chaque point $l = 1 \ldots n_k$:

- Le temps d'arrivée du véhicule, $t^+_{kl} = h_{ij}(t^-_{k,l-1})$ où $i = v_{r^{\pm}_{k,l-1}}$ et $j = v_{r^{\pm}_{kl}}$;
- Le temps de départ du véhicule, $t^-_{kl} = \max(t^+_{kl} + u_{r^{\pm}_{kl}},\ h^{-1}_{ij}(b_{r^{\pm}_{k,l+1}}))$ où $i = v_{r^{\pm}_{kl}}$ et $j = v_{r^{\pm}_{k,l+1}}$;
- La somme des temps de parcours jusqu'au point l, $w^{\Sigma}_{kl} = w^{\Sigma}_{k,l-1} + h_{ij}(t^-_{k,l-1}) - t^-_{k,l-1}$ où $i = v_{r^{\pm}_{k,l-1}}$ et $j = v_{r^{\pm}_{kl}}$;
- La somme des temps de retard jusqu'au point l, $x^{\Sigma}_{kl} = x^{\Sigma}_{k,l-1} + \max(0, t^+_{kl} - e_{r^{\pm}_{kl}})$;
- La capacité résiduelle du véhicule après avoir desservi le point l, $q_{kl} = q_{k,l-1} + q_{r^{\pm}_{kl}}$;
- Un booléen indiquant si la capacité du véhicule a été respectée jusqu'au point l, $q^{\geq 0}_{kl} = q^{\geq 0}_{k,l-1} \wedge q_{kl} \geq 0$.

Ces valeurs peuvent être calculées l'une à la suite de l'autre à partir de :

- $t^-_{k0} = \max(0,\ h^{-1}_{ij}(b_{r^{\pm}_{k1}}))$ où $i = v_0$ et $j = v_{r^{\pm}_{k1}}$;
- $w^{\Sigma}_{k0} = 0$;
- $x^{\Sigma}_{k0} = 0$;
- $q_{k0} = Q$;
- $q^{\geq 0}_{k0} =$ vrai.

Il y a une exception pour $t^-_{kn_k}$ qui doit être calculé de la manière suivante :

$$t^-_{kn_k} = \max(t^+_{kn_k} + u_{r^{\pm}_{kn_k}},\ \min(T', h^{-1}_{ij}(T)))$$

où $i = v_{r^{\pm}_{kn_k}}$ et $j = v_0$.

L'objectif défini à la Section 3.4 peut maintenant être réécrit sous la forme suivante :

$$\min \sum_{k=1}^{m} w^{\Sigma}_{kn_k} + (h_{ij}(t^-_{kn_k}) - t^-_{kn_k}) + \alpha x^{\Sigma}_{kn_k} + \beta \max(0, h_{ij}(t^-_{kn_k}) - T)$$

où $i = v_{r^{\pm}_{kn_k}}$ et $j = v_0$.

L'avantage de procéder de cette façon est que, lorsqu'une route dans la solution est modifiée, que ce soit par le retrait, l'ajout ou le déplacement d'un ou plusieurs points de service, on n'a qu'à mettre à jour ces valeurs à partir du premier changement. Il est cependant inévitable de devoir les recalculer jusqu'à la fin de la route puisqu'un seul changement peut avoir un effet domino sur les temps de parcours et les retards des points de service suivants.

4.5. Insertion des nouvelles requêtes

Lorsqu'une nouvelle requête doit être ajoutée à une solution existante, on énumère toutes les insertions possibles qui respectent la capacité des véhicules et l'on choisit celle de plus faible coût. Il faut noter que les points de service $r_{kl}^{\pm}$ qui sont fixés dans la solution, soit parce qu'ils ont déjà été desservis ou qu'ils correspondent à la destination courante d'un véhicule, peuvent être facilement identifiés avec la condition $t_{k,l-1}^{-} < \tau$, où $\tau \in [0, +\infty[$ est le temps présent. Il n'est évidemment pas possible d'insérer la nouvelle requête avant l'un de ces points.

L'Algorithme 4 détaille le processus d'insertion où :

- $l_k = \min\{l \in \{1, \ldots, n_k + 1\} \mid t_{k,l-1}^{-} \geq \tau\}$ est l'indice du premier point de service dans la route du véhicule k devant lequel il est possible d'insérer la nouvelle requête;
- Les valeurs présentées à la Section 4.4 sont mises à jour lors de chaque insertion en s'assurant que t_{k,l_k-1}^{-} a une valeur minimale de τ puisqu'un départ avant le temps présent est impossible;
- $z = \sum_{k=1}^{m} w_{kn_k}^{\Sigma} + (h_{ij}(t_{kn_k}^{-}) - t_{kn_k}^{-}) + \alpha x_{kn_k}^{\Sigma} + \beta \max(0, h_{ij}(t_{kn_k}^{-}) - T)$ où $i = v_{r_{kn_k}^{\pm}}$ et $j = v_0$, est la valeur de l'objectif pour la solution courante.

Pour chaque véhicule (ligne 5), on insère le point de collecte le plus tôt possible suivi immédiatement du point de livraison en mettant à jour la séquence des points de service (ligne 6), le nombre de points de service (ligne 7) et l'indice des nouveaux points de service dans la séquence (lignes 8 et 9). Si l'insertion respecte la capacité du véhicule et que le coût de la nouvelle solution est le plus faible jusqu'à présent (ligne 10), on garde en mémoire l'indice du véhicule (ligne 11) et des points de service (lignes 12 et 13) ainsi que le coût de la solution (ligne 14). On fait ensuite avancer le point de livraison d'une position à la fois (lignes 17 et 18) jusqu'à ce que celui-ci atteigne la fin de la route. On fait alors avancer le point de collecte d'une position et l'on place le point de livraison immédiatement après celui-ci (lignes 20-22). Lorsque les deux points de service sont à la fin de la route, on les retire (ligne 28). À chaque mouvement, on vérifie si la capacité est respectée et si la nouvelle solution a le plus faible coût jusqu'à présent (lignes 23-27). Une fois que cela est fait pour

Algorithme 4 : Insertion de la requête $r = (r^+, r^-)$.

1 $z^* \longleftarrow +\infty$
2 $k^* \longleftarrow -1$
3 $l^{+*} \longleftarrow -1$
4 $l^{-*} \longleftarrow -1$
5 **pour** $k = 1 \dots m$ **faire**
6 $\quad r^{\pm}_{kl_k}, \dots, r^{\pm}_{k,n_k+2} \longleftarrow r^+, r^-, r^{\pm}_{kl_k}, \dots, r^{\pm}_{kn_k}$
7 $\quad n_k \longleftarrow n_k + 2$
8 $\quad l^+ \longleftarrow l_k$
9 $\quad l^- \longleftarrow l_k + 1$
10 $\quad$ **si** $q^{\geq 0}_{kn_k} \wedge z < z^*$ **alors**
11 $\quad\quad k^* \longleftarrow k$
12 $\quad\quad l^{+*} \longleftarrow l^+$
13 $\quad\quad l^{-*} \longleftarrow l^-$
14 $\quad\quad z^* \longleftarrow z$
15 $\quad$ **tant que** $l^+ < n_k - 1$ **faire**
16 $\quad\quad$ **si** $l^- < n_k$ **alors**
17 $\quad\quad\quad r^{\pm}_{kl^-} \longleftrightarrow r^{\pm}_{k,l^-+1}$
18 $\quad\quad\quad l^- \longleftarrow l^- + 1$
19 $\quad\quad$ **sinon**
20 $\quad\quad\quad r^{\pm}_{kl^+}, \dots, r^{\pm}_{kn_k} \longleftarrow r^{\pm}_{k,l^+ +1}, r^+, r^-, r^{\pm}_{k,l^+ +2}, \dots, r^{\pm}_{k,l^- -1}$
21 $\quad\quad\quad l^+ \longleftarrow l^+ + 1$
22 $\quad\quad\quad l^- \longleftarrow l^+ + 1$
23 $\quad\quad$ **si** $q^{\geq 0}_{kn_k} \wedge z < z^*$ **alors**
24 $\quad\quad\quad k^* \longleftarrow k$
25 $\quad\quad\quad l^{+*} \longleftarrow l^+$
26 $\quad\quad\quad l^{-*} \longleftarrow l^-$
27 $\quad\quad\quad z^* \longleftarrow z$
28 $\quad n_k \longleftarrow n_k - 2$
29 $r^{\pm}_{k^*l_{k^*}}, \dots, r^{\pm}_{k^*,n_{k^*}+2} \longleftarrow$
$\quad r^{\pm}_{k^*l_{k^*}}, \dots, r^{\pm}_{k^*,l^{+*}-1}, r^+, r^{\pm}_{k^*l^{+*}}, \dots r^{\pm}_{k^*,l^{-*}-2}, r^-, r^{\pm}_{k^*,l^{-*}-1}, \dots, r^{\pm}_{k^*n_{k^*}}$
30 $n_{k^*} \longleftarrow n_{k^*} + 2$

tous les véhicules, on insère définitivement les deux points de service au meilleur endroit trouvé, ce qui produit la nouvelle solution courante (lignes 29 et 30).

Notons que l'insertion des deux points de service à la fin d'une route sera toujours réalisable puisqu'il n'y a aucune limite de retard, que le véhicule est vide à ce moment-là et que $q_r \leq Q$ pour tout $r \in R$. L'algorithme devrait donc toujours être en mesure de réaliser l'insertion.

Puisque l'ensemble des routes peut contenir un maximum de $2|R|$ points de service, le nombre de combinaisons possibles pour l'insertion du point de collecte et du point de livraison est de l'ordre de $O(|R|^2)$. Pour chaque solution générée par l'une de ces combinaisons, les valeurs servant à l'évaluation du coût sont recalculées pour chaque point de service de la route à partir de l'endroit de l'insertion, soit un maximum de $2|R|$ fois. À chacune de ces fois, un temps de parcours est obtenu en consultant une fonction $h_{ij}(t)$. Comme nous avons vu à la Section 4.2.4, la complexité de la consultation est de l'ordre de $O(\log |PR_{h_{ij}}|)$ où $PR_{h_{ij}}$ est l'ensemble des points de rupture de la fonction h_{ij}. Dans le pire des cas, la complexité de l'algorithme d'insertion sera donc de l'ordre de $O(|R|^3 \log(\max_{i,j \in V} |PR_{h_{ij}}|))$.

4.6. Implémentation de la recherche tabou

Dans cette section, nous présentons de manière détaillée l'implémentation de notre méthode de recherche incluant la création d'une solution de départ, la définition de la structure de voisinage et du critère tabou, le déroulement d'une itération et l'utilisation de phases de diversification.

4.6.1. Solution initiale

Le processus d'optimisation commence par initialiser l'ensemble des requêtes actives R' avec les requêtes statiques qui sont connues au début de la journée, c'est-à-dire $R' = \{r \in R \mid t_r = 0\}$

Une solution initiale est ensuite construite en insérant de façon séquentielle chacune des requêtes $r \in R'$. À chaque itération, la requête ayant le meilleur coût d'insertion parmi toutes les requêtes restantes (c'est-à-dire celles ne faisant pas encore partie des routes courantes) est sélectionnée et insérée dans la solution. Cette procédure est répétée jusqu'à ce que toutes les requêtes soient incluses dans la solution qui devient alors la solution de départ pour la recherche tabou. Par la suite, les requêtes dynamiques seront insérées dans la solution au fur et à mesure qu'elles deviendront connues.

4.6.2. Structure de voisinage

La recherche tabou améliore la solution qui lui est fournie en utilisant une structure de voisinage, ce qui correspond à une classe de modifications appliquées à cette solution afin d'en obtenir de nouvelles qui ne sont pas trop différentes. Plusieurs classes de modifications existent dont certaines sont très simples et d'autres très complexes. Nous utiliserons ici un

voisinage assez simple. En plus de demander une implémentation moins élaborée qu'une structure de voisinage plus complexe, nous pourrons évaluer ce voisinage plus rapidement et ainsi effectuer plus d'itérations en un temps donné, ce qui est particulièrement important dans un contexte dynamique.

Notre voisinage sera donc constitué de toutes les solutions réalisables que l'on peut obtenir en retirant les points de service d'une requête quelconque de la solution courante et en les réinsérant à des positions quelconques dans la même route ou dans une autre route, de façon à ce qu'au moins un point de service change de position.

Notons qu'aucun point de service ayant déjà été desservi ne peut être déplacé. Par ailleurs, un point de livraison non encore desservi dont le point de collecte correspondant a déjà été desservi ne peut être déplacé qu'à une autre position à l'intérieur de la même route.

4.6.3. Critère tabou

Le critère tabou que nous utilisons pour identifier les modifications interdites au cours des prochaines itérations correspond aux points de service dont la réinsertion a généré la meilleure solution dans le voisinage. Plus précisément, pour les λ prochaines itérations, toute solution obtenue par une nouvelle réinsertion de ces points de service sera ignorée. Ceci a pour but d'empêcher le retour à une solution déjà rencontrée en interdisant les modifications inverses à celles appliquées récemment.

4.6.4. Itération

L'Algorithme 5 présente le déroulement d'une itération de la recherche tabou où S est la solution courante, S^* est la meilleure solution trouvée jusqu'à présent et z^* est le coût de S^*.

Pour chaque requête $r \in R'$ (ligne 3), on commence par retirer les points de service correspondants qui n'ont pas encore été desservis et ne sont pas la destination courante d'un véhicule en route (lignes 8 et 11). Si aucun des deux points de service ne satisfait cette condition, on considère alors que la requête n'est pas admissible et on la retire de R' (ligne 23). Dans le cas où seul le point de livraison satisfait la condition, celui-ci ne peut être que déplacé dans la même route.

Afin d'explorer toutes les solutions dans le voisinage, il faut identifier la meilleure réinsertion possible pour chacune des requêtes actives, et conserver en mémoire la meilleure solution obtenue de cette façon. Pour les problèmes avec un grand nombre de requêtes, un temps

Algorithme 5 : Itération

```
1  S' ⟵ Nul
2  z' ⟵ +∞
3  pour r = (r⁺, r⁻) ∈ R' faire
4      S'' ⟵ S
5      k ⟵ Indice de la route desservant r dans la solution S''
6      l⁻ ⟵ Position de r⁻ dans la route k de la solution S''
7      si t_{k,l⁻−1} ≥ τ alors
8          Retirer r⁻ de la route k dans la solution S''
9          l⁺ ⟵ Position de r⁺ dans la route k de la solution S''
10         si t_{k,l⁺−1} ≥ τ alors
11             Retirer r⁺ de la route k dans la solution S''
12         sinon
13             l⁺ ⟵ −1
14         z ⟵ Coût de S''
15         si z < z' alors
16             S' ⟵ S''
17             z' ⟵ z
18             r' = (r'⁺, r'⁻) ⟵ r = (r⁺, r⁻)
19             k' ⟵ k
20             l⁺' ⟵ l⁺
21             l⁻' ⟵ l⁻
22     sinon
23         R' ⟵ R' \ {r}
24 si S' ≠ Nul alors
25     si l⁺' > 0 alors
26         Insérer r'⁺ et r'⁻ dans S', si possible de façon à ce qu'ils ne soient pas aux
             positions l⁺' et l⁻' de la route k'
27     sinon
28         Insérer r'⁻ dans la route k' de la solution S', si possible de façon à ce qu'il ne
             soit pas à la position l⁻'
29     Ajouter un tabou sur la requête r' pour une durée de λ itérations
30     S ⟵ S'
31     z ⟵ Coût de S
32     si z < z* alors
33         S* ⟵ S
34         z* ⟵ z
```

d'exécution considérable peut être requis puisque, rappelons-le, chaque insertion a une complexité dans l'ordre de $O(|R|^3 \log(\max_{i,j \in V} |PR_{h_{ij}}|))$. Pour réduire le temps de calcul, seule la requête dont le retrait a mené à la plus grande réduction du coût de la solution (lignes

15-21) sera considérée pour la réinsertion (lignes 25-28). Cette approche peut évidemment produire une solution de moins bonne qualité que si l'on considérait toutes les réinsertions possibles, mais permet d'effectuer plus d'itérations en un temps donné.

La réinsertion des points de service est effectuée à l'aide d'une méthode semblable à la méthode d'insertion présentée à la Section 4.5, sauf qu'on ne peut réinsérer la requête à sa position originale. Dans le cas où seulement le point de livraison a pu être retiré, on doit également limiter sa réinsertion à la route où il se trouvait déjà puisque le point de collecte correspondant a déjà été desservi par ce véhicule (ligne 28).

On introduit ensuite un tabou sur la requête dont la réinsertion a mené à la meilleure solution (ligne 29). Essentiellement, cela veut dire que cette requête est retirée de R' immédiatement et y sera réintroduite lorsque λ itérations de plus auront été réalisées.

Finalement, la nouvelle solution ainsi générée devient la solution courante pour la prochaine itération de la recherche tabou (ligne 30). De plus, si son coût est inférieur à celui de la meilleure solution rencontrée jusqu'à maintenant, on la conserve en mémoire, car elle devient la nouvelle meilleure solution (lignes 32-34).

Dans certains cas, comme lorsqu'il ne reste qu'un point de livraison à desservir dans une route, il peut cependant être impossible de réinsérer les points de service à des positions différentes. Nous les réinsérons alors à leurs positions originales. La solution courante demeure donc la même pour cette itération, mais puisqu'un tabou est tout de même ajouté sur la requête, la prochaine itération retirera une requête différente et de nouvelles solutions pourront être explorées.

4.6.5. Diversification

Puisque le voisinage utilisé contient des solutions assez semblables à la solution courante, échapper à un minimum local peut être difficile malgré le recours au critère tabou. Afin d'explorer de nouvelles régions dans l'espace des solutions, nous introduisons des phases de diversification de manière périodique lors de la recherche. Celles-ci surviennent à toutes les ρ itérations et consistent en σ itérations consécutives où une requête $r \in R'$, choisie de manière aléatoire, est retirée de la solution courante et réinsérée également de manière aléatoire.

Notons que les tabous en place se terminent immédiatement lorsqu'une phase de diversification débute et qu'aucun tabou n'est associé aux requêtes choisies de manière aléatoire.

Chapitre 5

Tests numériques

Dans ce chapitre, nous commençons par présenter les données, la procédure et les paramètres que nous utilisons pour tester notre méthode de résolution. Ensuite, nous présentons les résultats obtenus afin d'évaluer la performance de celle-ci ainsi que l'impact du dynamisme, de l'exploration du voisinage complet, de la diversification et de la structure DSP.

5.1. Données pour les tests

Afin de tester notre méthode de résolution et d'évaluer son efficacité, nous avons créé plusieurs instances de notre problème à partir des instances NEWLET provenant des travaux de Ben Ticha [**25**] (également utilisées par Gmira et al. [**10**]) pour le problème de tournées de véhicules avec fenêtres de temps sur un réseau routier avec variations de vitesse sur les arcs et requêtes constituées d'un seul point de service. Les réseaux utilisés par ces instances ont été générés par la méthode de Letchford et al. [**16**] qui consiste à placer des sommets de manière aléatoire sur un plan à deux dimensions et de connecter ensuite les sommets les plus rapprochés, en évitant les croisements, afin d'obtenir un réseau aussi réaliste que possible.

La taille de ces instances varie de 16 requêtes sur un réseau de 50 sommets à 200 requêtes sur un réseau de 500 sommets.

Comme nous devons résoudre un problème avec des requêtes deux-points, les requêtes sont regroupées aléatoirement deux par deux. La requête dont la fenêtre de temps se termine en premier devient le point de collecte et l'autre le point de livraison. Lorsque le nombre de requêtes est impair, une requête est simplement mise de côté. On se retrouve donc avec des instances de tailles variant de 8 requêtes (16 points de service) sur un réseau de 50 sommets à 100 requêtes (200 points de service) sur un réseau de 500 sommets.

Pour obtenir des instances dynamiques, nous avons également généré un temps d'arrivée t_r aléatoire selon une distribution uniforme entre 0 et b_{r^+} pour 50% des requêtes $r = (r^+, r^-) \in R$. Les autres (requêtes statiques) ont une heure d'arrivée $t_r = 0$. Nos instances ont donc un taux de dynamisme de 50%.

Les temps sont représentés par une valeur continue non négative. Le début de la journée est au temps 0, la fin de la journée au temps $T = 100$ et des changements de vitesse surviennent sur les arcs aux temps 20, 30, 70 et 80.

Pour toutes les instances, le temps maximal pour retourner au dépôt lorsqu'un véhicule est en attente a été fixé à $T' = \dfrac{\max_{r \in R}(t_r) + T}{2}$. La capacité des véhicules est $Q = 200$.

Le Tableau 1 présente le nombre d'instances dont nous disposons en fonction du nombre de sommets et de requêtes.

Sommets \ Requêtes	8	12	16	25	33	37	50	66	100	Tous
50	30	-	30	-	-	-	-	-	-	60
100	-	30	30	30	30	-	-	-	-	120
200	-	30	-	30	30	30	30	30	-	180
500	-	-	-	-	-	-	-	-	30	30
Tous	30	60	60	60	60	30	30	30	30	390

Tableau 1. Nombre d'instances par taille

5.2. Procédure de simulation

Afin de simuler une situation où la résolution du problème se produit en même temps que l'exécution des routes et l'arrivée des nouvelles requêtes, nous procédons de la manière suivante.

Tout d'abord, la structure DSP donnant le temps de parcours pour chaque paire de sommets du réseau est calculée à l'aide de la méthode présentée à la Section 4.2 avec les temps de départ 0, 20, 30, 70, 80 et 100.

Ensuite, on démarre le temps. Un facteur de conversion entre le temps réel et celui du problème à résoudre est établi. Comme nous avons un bon nombre d'instances à tester et que celles-ci ne sont pas de très grande taille, nous avons opté pour de très courtes journées simulées où chaque unité de temps correspond à 1 seconde. Puisque T = 100, nous avons donc des simulations dont la durée est d'environ 100 secondes. D'autres facteurs de conversion ont également été testés, mais n'ont eu qu'un impact minime sur les résultats observés.

On construit ensuite notre solution initiale avec les requêtes statiques tel que présenté à la Section 4.6.1. Les véhicules vont suivre cette solution au début tandis que la recherche tabou est lancée afin de l'optimiser. Chaque fois que la recherche tabou identifie une meilleure solution, cette dernière remplace alors celle qui est actuellement suivie par les véhicules. Cette substitution ne pose aucune difficulté étant donné que les points de service qui ont déjà été desservis ou qui correspondent à la prochaine destination d'un véhicule ne peuvent être déplacés lors de l'optimisation. Par ailleurs, une interruption est programmée pour l'arrivée de la prochaine requête à l'aide d'un objet de type *Timer*. Dans une situation réelle, cette interruption proviendrait plutôt d'un système extérieur qui reçoit les requêtes des clients. Lorsqu'elle survient, la recherche tabou s'arrête. La nouvelle requête est alors insérée dans la meilleure solution suivie par les véhicules et cette solution est ensuite utilisée pour relancer l'optimisation avec la recherche tabou.

Il faut noter qu'à chaque itération de la recherche tabou, une vérification est effectuée afin de déterminer si un ou plusieurs véhicules ont atteint leur destination courante et se dirigent actuellement vers une nouvelle destination dans la meilleure solution. Dans un tel cas, les nouvelles destinations sont fixées et ne peuvent plus être déplacées. Également, comme la solution courante de la recherche tabou n'est peut-être plus cohérente avec la meilleure solution suivie par les véhicules, cette meilleure solution devient la solution courante dans la recherche tabou, avant qu'elle ne poursuive l'optimisation. Dans un tel cas, les requêtes ayant un statut tabou redeviennent non taboues.

La Figure 6 présente un exemple illustrant cette dernière situation. Alors que le véhicule se dirige vers le point de service A dans la meilleure solution, la recherche tabou échange l'ordre de visite des points de service B et C dans la solution courante. Si l'on suppose que cet ordre de visite des points B et C n'est pas modifié par la suite par la recherche tabou et qu'aucune nouvelle meilleure solution n'est identifiée, une incohérence surviendra. En effet, lorsque le véhicule aura atteint sa destination courante dans la meilleure solution, soit le point de service A, il se dirigera ensuite vers le point de service B, ce qui est incompatible avec la visite de C dans la solution courante. Cette dernière devra donc être remplacée.

5.3. Paramètres de résolution

Pour évaluer le coût des solutions lors de la résolution, nous multiplions les temps de retard aux points de service par $\alpha = 5$ et les temps de retard au dépôt par $\beta = 10$.

Le nombre ρ d'itérations entre chaque phase de diversification, le nombre σ d'itérations que comprennent ces phases et le nombre λ d'itérations pendant lesquelles les tabous restent

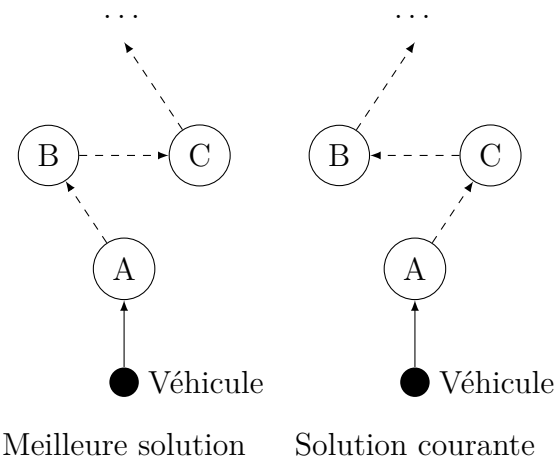

Figure 6. Exemple d'incohérence entre la meilleure solution et la solution courante

effectifs changent quant à eux au cours de la résolution en fonction du nombre de points de service pouvant être déplacés par la procédure d'optimisation au temps présent $\theta = \sum_{k=1}^{m} |\{l \in \{1, \ldots, n_k + 1\} \mid t^{-}_{k,l-1} \geq \tau\}|$. À tout moment, on a ainsi $\rho = 10\theta$, $\sigma = \frac{1}{2}\theta$ et $\lambda = \frac{3}{8}\theta$.

Ces valeurs pour les paramètres ont été retenues à la suite des résultats obtenus lors de tests préliminaires.

5.4. Résultats

Tous les résultats présentés dans cette section ont été obtenus suite à l'exécution d'un programme développé avec le langage Java sur un système doté d'un processeur Intel Core i5-4670k opérant à une fréquence de 3.4 GHz et de 16 Go de mémoire.

5.4.1. Performance de la recherche tabou

Une façon d'évaluer la performance de notre recherche tabou est de comparer le coût des solutions obtenues par notre méthode à celles obtenues avec une approche vorace qui ne fait qu'insérer les requêtes à la meilleure position possible lors de leur arrivée.

Le Tableau 2 présente l'amélioration moyenne du coût de la solution selon le nombre de requêtes pour trois variantes de nos instances tests. La première de ces variantes a accès à un nombre illimité de véhicules $m = +\infty$ comme c'est le cas dans la version originale des instances NEWLET. Nous avons ensuite décidé de tester deux variantes où le nombre de véhicules est limité en fonction du nombre de requêtes à desservir, soit $m = \left\lfloor \frac{|R|}{2} \right\rfloor$ et

$m = \left\lfloor \frac{|R|}{4} \right\rfloor$. Notons que, puisque les fenêtres de temps sont souples et qu'il n'y a aucune limite de retard aux points de service et au dépôt, toutes les requêtes peuvent être desservies même si le nombre de véhicules est limité.

Nombre de requêtes	$m = +\infty$	$m = \left\lfloor \frac{\lvert R\rvert}{2} \right\rfloor$	$m = \left\lfloor \frac{\lvert R\rvert}{4} \right\rfloor$
8	3.89%	5.39%	19.60%
12	3.01%	4.55%	13.42%
16	4.98%	7.96%	18.13%
25	4.53%	7.33%	16.03%
33	5.63%	5.97%	19.48%
37	7.53%	8.30%	19.21%
50	6.33%	6.19%	19.36%
66	6.74%	7.58%	17.43%
100	7.87%	8.30%	13.53%
Tous	5.28%	6.72%	17.17%

Tableau 2. Pourcentage moyen d'amélioration des solutions produites par la recherche tabou

Comme on peut le constater au Tableau 2, plus le nombre de véhicules est limité, plus il est difficile d'obtenir une solution de bonne qualité en effectuant seulement des insertions et plus la recherche tabou devient utile.

Les Tableaux 3, 4 et 5 présentent respectivement le nombre moyen de véhicules utilisés, le temps de retard moyen aux points de service et le temps de retard moyen au dépôt à la fin de la journée dans les solutions trouvées par la méthode d'insertion (colonnes I) et par la méthode de recherche tabou (colonnes RT) pour chacune des trois variantes.

Nombre de requêtes	$m = +\infty$		$m = \left\lfloor \frac{\lvert R\rvert}{2} \right\rfloor$		$m = \left\lfloor \frac{\lvert R\rvert}{4} \right\rfloor$	
	I	RT	I	RT	I	RT
8	3.47	3.47	3.27	3.27	2	2
12	6.58	6.50	5.67	5.63	3	3
16	7.58	7.62	6.87	6.87	4	4
25	12.77	12.60	11.25	11.22	6	6
33	15.88	16.18	14.67	14.77	8	8
37	18.43	18.93	17.07	17.03	9	9
50	22.60	23.03	22.07	22.60	12	12
66	28.80	29.60	28.40	29.20	16	16
100	37.80	38.90	36.47	37.97	24.4	25
Tous	15.13	15.36	14.17	14.39	8.11	8.15

Tableau 3. Nombre moyen de véhicules utilisés

Nombre de requêtes	$m = +\infty$		$m = \left\lfloor \frac{\lvert R \rvert}{2} \right\rfloor$		$m = \left\lfloor \frac{\lvert R \rvert}{4} \right\rfloor$	
	I	RT	I	RT	I	RT
8	0.80	0.79	1.28	1.16	4.82	3.24
12	3.34	3.21	4.46	4.17	13.94	11.92
16	1.97	1.88	2.81	2.51	8.09	6.55
25	3.78	3.61	4.62	4.20	12.49	10.47
33	3.69	3.40	4.14	3.80	11.11	8.97
37	4.41	3.92	4.89	4.36	12.57	10.24
50	4.09	3.67	4.21	3.82	11.05	8.84
66	4.19	3.76	4.26	3.76	9.45	7.79
100	4.40	4.12	4.61	4.30	8.82	7.60
Tous	3.34	3.11	3.95	3.60	10.61	8.73

Tableau 4. Retard moyen aux points de service

Nombre de requêtes	$m = +\infty$		$m = \left\lfloor \frac{\lvert R \rvert}{2} \right\rfloor$		$m = \left\lfloor \frac{\lvert R \rvert}{4} \right\rfloor$	
	I	RT	I	RT	I	RT
8	0.05	0.05	0.39	0.11	4.03	2.45
12	2.64	2.64	4.39	4.07	23.06	19.36
16	0.96	0.86	1.97	1.49	11.10	8.85
25	2.14	2.06	3.40	2.94	19.05	14.63
33	2.04	1.92	2.71	2.58	16.41	11.94
37	2.91	2.47	3.63	3.22	18.55	13.78
50	2.76	2.44	2.90	2.52	15.67	11.48
66	2.47	2.18	2.57	2.21	12.30	9.19
100	5.22	4.78	5.75	5.30	15.59	13.73
Tous	2.23	2.07	3.09	2.73	15.80	12.32

Tableau 5. Retard moyen au dépôt

Pour les prochains tests numériques, nous utiliserons uniquement la version originale des instances où le nombre de véhicules est illimité ($m = +\infty$).

5.4.2. Impact du dynamisme

Afin d'évaluer l'impact du dynamisme, le Tableau 6 compare le coût des solutions produites dans le contexte dynamique avec le coût des solutions produites avec une information parfaite, c'est-à-dire en supposant que toutes les requêtes sont connues dès le début de la journée. Pour ce faire, nous présentons le pourcentage d'augmentation du coût de nos solutions par rapport à celles obtenues avec l'information parfaite. Comme on peut le constater, l'impact du dynamisme est important avec une augmentation moyenne de près de 20%. Ceci indique les bénéfices potentiels qui pourraient être obtenus avec une approche qui tenterait

de prévoir les requêtes futures (en supposant que l'on dispose d'une distribution de probabilités), plutôt que de simplement réagir à chacune des nouvelles requêtes qui sont reçues.

Nombre de requêtes	**Augmentation**
8	12.98%
12	20.44%
16	16.20%
25	22.76%
33	20.42%
37	20.43%
50	19.97%
66	19.06%
100	8.65%
Tous	18.52%

Tableau 6. Pourcentage moyen d'augmentation du coût de la solution dû au dynamisme

5.4.3. Impact de l'exploration du voisinage complet

À la Section 4.6.4, nous avons expliqué que lors de l'exploration du voisinage de la solution courante, nous nous sommes contentés de considérer la réinsertion de la requête dont le retrait apporte la plus grande réduction du coût de la solution. Cette approche permet de réaliser davantage d'itérations en un laps de temps donné. Par contre, nous n'identifions pas nécessairement la meilleure solution dans le voisinage.

À cet effet, le Tableau 7 présente le pourcentage d'augmentation du coût de la solution lorsque l'on consacre plus de temps de calcul à chacune des itérations de la recherche tabou afin d'identifier la meilleure solution dans le voisinage. En fait, on constate que cette augmentation est peu élevée (moyenne de 0.22%) et mène en fait dans bien des cas à une réduction du coût. Il reste que, de façon générale, un plus grand nombre d'itérations demandant moins de temps de calcul semble améliorer légèrement la qualité des solutions.

5.4.4. Impact de la diversification

Le Tableau 8 présente maintenant le pourcentage d'augmentation du coût des solutions si l'on retire la phase de diversification de la recherche tabou, telle que présentée à la Section 4.6.5. Encore une fois, cette augmentation n'est pas énorme, mais tout de même présente dans la plupart des cas à l'exception des instances de 100 requêtes. L'ajout de la phase de diversification apparaît donc bénéfique pour la recherche tabou.

Nombre de requêtes	Augmentation
8	-0.01%
12	-0.18%
16	0.21%
25	0.29%
33	-0.20%
37	0.87%
50	1.48%
66	-0.01%
100	0.26%
Tous	0.22%

Tableau 7. Pourcentage moyen d'augmentation du coût de la solution dû à l'exploration du voisinage complet

Nombre de requêtes	Augmentation
8	0.37%
12	0.21%
16	0.52%
25	0.81%
33	0.41%
37	1.26%
50	1.24%
66	0.39%
100	-0.51%
Tous	0.51%

Tableau 8. Pourcentage moyen d'augmentation du coût de la solution dû à l'absence de diversification

5.4.5. Impact de la structure DSP

5.4.5.1. *Construction de la structure DSP*

Comme on peut l'imaginer, le temps requis pour construire la structure DSP ainsi que la quantité de données contenues dans celle-ci sont fortement dépendants de la taille du réseau. Le Tableau 9 présente le temps de calcul moyen ainsi que la taille moyenne de la structure complète des 60 réseaux utilisés dans nos instances en fonction du nombre de sommets (15 réseaux pour chaque taille de 50, 100, 200 et 500 sommets). On peut observer que l'évolution de ces caractéristiques de la structure DSP est d'un ordre inférieur à $O(|V|^3)$.

Dans le cas d'instances de grande taille, il pourrait être possible de réduire le temps de calcul initial ainsi que la taille de la structure en ne calculant les chemins les plus rapides que pour les paires de sommets où il y a des requêtes plutôt que pour toutes les paires de sommets

Nombre de sommets	Temps de calcul moyen (s)	Taille moyenne (Mo)
50	0.09	1.35
100	0.43	7.01
200	2.94	37.59
500	39.61	357.23

Tableau 9. Temps de calcul moyen et taille moyenne de la structure DSP

du réseau comme nous l'avons fait dans nos tests. Par contre, comme on ne connaît pas les sommets où des requêtes arriveront de manière dynamique, un certain temps de résolution devrait alors être sacrifié par la suite pour calculer les meilleurs chemins entre ces nouveaux sommets.

Le Tableau 10 présente certaines statistiques additionnelles sur les chemins produits. La colonne *Nombre de chemins* contient le nombre moyen de chemins différents trouvés pour chaque paire de sommets du réseau, la colonne *Nombre d'arcs* contient le nombre moyen d'arcs qui constituent ces chemins et la colonne *Réduction* contient le pourcentage moyen de réduction des temps de parcours par rapport aux chemins trouvés par l'algorithme de Dijkstra en considérant un temps de parcours fixe, non dépendant du moment de la journée, sur chaque arc du réseau. Plus précisément, le temps de parcours sur un arc correspond alors à la moyenne des temps de parcours variables sur cet arc. Notons toutefois que les chemins ainsi obtenus sont tout de même évalués avec les temps de parcours variables afin de calculer le pourcentage de réduction.

Nombre de sommets	Nombre de chemins	Nombre d'arcs	Réduction
50	1.28	6.76	8.29%
100	1.41	9.08	10.66%
200	1.59	12.47	12.80%
500	1.85	18.48	15.05%

Tableau 10. Statistiques sur les chemins

5.4.5.2. *Amélioration des solutions due à la structure DSP*

Le Tableau 11 présente maintenant le pourcentage moyen d'amélioration que la structure DSP apporte au coût des solutions trouvées par notre méthode. En effet, avec des temps de parcours plus rapides, on observe également moins de retard aux points de service et au dépôt, ce qui mène à une amélioration importante de près de 30% en moyenne au coût des solutions.

Les retards moyens sont présentés en détail au Tableau 12. Dans ce tableau, les colonnes 1C correspondent aux solutions obtenues en utilisant les chemins trouvés avec le temps de

Nombre de requêtes	Amélioration
8	17.88%
12	32.37%
16	21.43%
25	29.85%
33	30.42%
37	35.40%
50	34.28%
66	31.51%
100	30.98%
Tous	29.09%

Tableau 11. Pourcentage moyen d'amélioration du coût de la solution dû à la structure DSP

parcours moyen de chaque arc et les colonnes DSP correspondent aux solutions obtenues en utilisant la structure DSP.

Nombre de requêtes	Retard aux points de service			Retard au dépôt		
	1C	DSP	Réduction	1C	DSP	Réduction
8	1.30	0.79	38.93%	0.38	0.05	86.98%
12	5.24	3.21	38.65%	5.72	2.64	53.76%
16	2.89	1.88	35.11%	1.75	0.86	50.94%
25	5.60	3.61	35.58%	4.67	2.06	55.90%
33	5.35	3.40	36.41%	4.67	1.92	58.89%
37	6.62	3.92	40.74%	6.52	2.47	62.06%
50	5.88	3.67	37.48%	5.85	2.44	58.24%
66	5.83	3.76	35.60%	5.41	2.18	59.74%
100	6.53	4.12	36.86%	7.97	4.78	40.10%
Tous	4.95	3.11	37.08%	4.59	2.07	55.01%

Tableau 12. Réduction des temps de retard moyens due à la structure DSP

Chapitre 6

Conclusion

Pour commencer, nous avons exploré différents travaux portant sur des problèmes ayant des aspects en commun avec le nôtre tels que le dynamisme, les temps de parcours variables et les requêtes deux-points.

Nous avons ensuite formulé notre problème sur un réseau routier où les sommets représentent des intersections et les arcs représentent des segments d'arcs routiers ayant une vitesse de circulation qui varie selon le moment de la journée, où les requêtes ont un point de collecte et un point de livraison, et où l'objectif est de minimiser le temps total de parcours des véhicules et les temps de retard aux points de service et au dépôt.

Par la suite, nous avons décrit notre méthode de résolution par recherche tabou qui s'inspire de quelques-uns des travaux rapportés dans la littérature. Nous utilisons la structure DSP pour calculer les temps de parcours en considérant différents chemins entre chaque paire de sommets et une structure de voisinage simple qui consiste à déplacer une requête à la fois pour explorer de nouvelles solutions. Afin d'ajouter de la variété dans les solutions visitées, nous associons un tabou à la requête déplacée à chaque itération et faisons appel à des phases de diversification de manière périodique.

Finalement, nous avons utilisé un total de 390 instances générées de manière synthétique afin d'évaluer l'efficacité de notre méthode. Le coût des solutions trouvées a une bonne amélioration par rapport à celles obtenues par de simples insertions, particulièrement lorsque le nombre de véhicules est limité. Nous avons également testé différentes variantes de la méthode où l'on possède l'information parfaite, où l'on explore le voisinage complet, où l'on n'utilise pas de diversification et où l'on ne considère qu'un seul chemin entre chaque paire de sommets. Cette dernière variante nous permet d'observer l'importante amélioration que la structure DSP apporte à nos solutions.

Bien que ces instances soient d'une taille inférieure à ce qui est généralement observé en pratique, le fait d'obtenir des améliorations importantes par rapport à une méthode d'insertion, et ce, en des temps de calcul très courts sur une machine qui n'est pas particulièrement puissante, nous incite à penser que notre méthode pourrait être viable dans une situation où l'on disposerait d'un réseau de plus grande taille avec un plus grand nombre de requêtes, en lui accordant davantage de temps de calcul.

L'application de notre méthode à des instances utilisant des données provenant d'un réseau routier réel serait d'ailleurs une piste intéressante à explorer lors de prochaines recherches. Certaines modifications et extensions au problème ainsi qu'à la méthode de résolution pourraient également être considérées telles que l'utilisation d'autres structures de voisinage, l'exploitation de données probabilistes quant à l'arrivée des requêtes ou encore la présence d'incertitude au niveau des temps de parcours.

Références bibliographiques

[1] Andrea Attanasio, Jay Bregman, Gianpaolo Ghiani, and Emanuele Manni. Real-time fleet management at ecourier ltd. In *Dynamic Fleet Management*, pages 219–238. Springer, 2007.

[2] Maria Battarra, Jean-François Cordeau, and Manuel Iori. Chapter 6: pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM, 2014.

[3] Alexandre Beaudry, Gilbert Laporte, Teresa Melo, and Stefan Nickel. Dynamic transportation of patients in hospitals. *OR spectrum*, 32(1):77–107, 2010.

[4] Ann Melissa Campbell and Martin WP Savelsbergh. Decision support for consumer direct grocery initiatives. *Transportation Science*, 39(3):313–327, 2005.

[5] Zhi-Long Chen and Hang Xu. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88, 2006.

[6] Bingbing Dan, Wanhong Zhu, Huabing Li, Yangyang Sang, and Yan Liu. Dynamic optimization model and algorithm design for emergency materials dispatch. *Mathematical Problems in Engineering*, 2013, 2013.

[7] Jan Fabian Ehmke and Dirk Christian Mattfeld. Vehicle routing for attended home delivery in city logistics. *Procedia-Social and Behavioral Sciences*, 39:622–632, 2012.

[8] Michel Gendreau, Francois Guertin, Jean-Yves Potvin, and René Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174, 2006.

[9] Michel Gendreau, Francois Guertin, Jean-Yves Potvin, and Eric Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, 33(4):381–390, 1999.

[10] Maha Gmira, Michel Gendreau, Andrea Lodi, and Jean-Yves Potvin. *Tabu Search for the Time-Dependent Vehicle Routing Problem with Time Windows on a Road Network*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise ..., 2019.

[11] Jiangchuan Huang and Raja Sengupta. Stability of dynamic traveling repairman problem under polling-sequencing policies. In *2013 European Control Conference (ECC)*, pages 614–619. IEEE, 2013.

[12] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34(4):426–438, 2000.

[13] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Vehicle dispatching with time-dependent travel times. *European journal of operational research*, 144(2):379–396, 2003.

[14] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225, 2006.

[15] Mostepha R Khouadjia, Briseida Sarasola, Enrique Alba, Laetitia Jourdan, and El-Ghazali Talbi. A comparative study between dynamic adapted pso and vns for the vehicle routing problem with dynamic requests. *Applied Soft Computing*, 12(4):1426–1439, 2012.

[16] Adam N Letchford, Saeideh D Nasiri, and Amar Oukil. Pricing routines for vehicle routing with time windows on road networks. *Computers & Operations Research*, 51:331–337, 2014.

[17] Quan Lu and Maged M Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175(2):672–687, 2006.

[18] Chryssi Malandraki and Mark S Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation science*, 26(3):185–200, 1992.

[19] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[20] Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

[21] Yi Qu and Timothy Curtois. Job insertion for the pickup and delivery problem with time windows. *Lecture Notes in Management Science*, 9:26–32, 2017.

[22] Mostafa Setak, Majid Habibi, Hossein Karimi, and Mostafa Abedzadeh. A time-dependent vehicle routing problem in multigraph with fifo property. *Journal of Manufacturing Systems*, 35:37–45, 2015.

[23] Eiichi Taniguchi and Hiroshi Shimamoto. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C: Emerging Technologies*, 12(3-4):235–250, 2004.

[24] Barrett W Thomas. Waiting strategies for anticipating service requests from known customer locations. *Transportation Science*, 41(3):319–331, 2007.

[25] Hamza Ben Ticha. *Vehicle Routing Problems with road-network information.* PhD thesis, Université Clermont Auvergne, 2017.