Planning and routing via decomposition approaches

Nastaran Rahmani

université de BORDEAUX

THÈSE

présentée pour obtenir le grade de

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

SPÉCIALITÉ : MATHÉMATIQUES APPLIQUÉES

par Nastaran RAHMANI

PLANNING AND ROUTING VIA DECOMPOSITION APPROACHES

Sous la direction de : François VANDERBECK

Soutenue le 26 juin 2014.

Membres du jury et invités:

M. FEILLET, Dominique	Professeur, École des Mines de Saint-Etienne	Président
M. HANAFI, Saïd	Professeur, Université de Valenciennes	Rapporteur
M. JOUGLET, Antoine	MCF, Université de Technologie de Compiègne	Rapporteur
M. CLAUTIAUX, François	Professeur, Université de Bordeaux	Examinateur
M. DETIENNE, Boris	MCF, Université de Bordeaux	Invité (coencadrant)
M. SADYKOV, Ruslan	Research Fellow, INRIA-Bordeaux	Invité (coencadrant)
M. VANDERBECK, François	Professeur, Université de Bordeaux	Directeur

Planification et Routage via les Approches de Décomposition

Résumé. Problèmes de tournées de véhicules statiques et déterministes ne peuvent pas être utilisés dans de nombreux systémes de la vie réelle, du fait que les données d'entrée ne sont pas fiables et sont révélées au fil du temps. Dans cette thèse, nous étudions un problème de ramassage et de livraison avec fenêtres de temps et un maximum de temps de trajet - le problème dial-a-ride - dans sa variante statique et dynamique, et nous faisons des propositions spécifiques sur les modèles d'optimisation robustes pour résoudre ce problème. Pour résoudre le modèle statique, nous développons une approche branch-and-price qui gère toutes les contraintes de temps dans le processus de création d'itinéraires de véhicules. Notre travail est axé sur les techniques de résolution du sous-problème et d'accélération pour l'approche branch-and-price. Nos résultats numériques montrent que la méthode est compétitive par rapport aux approches existantes qui sont basées sur le branch-and-cut. Dans le contexte dynamique, où certaines données d'entrée sont révélées dynamiquement ou modifiées au fil du temps, nous appliquons notre algorithme branch-and-price pour la ré-optimisation dans une approche sur horizon glissant.

Mot clés: dial-a-ride problème, programmation dynamique, branch-and-price, optimisation dynamique, optimisation robuste.

Planning and Routing via Decomposition Approaches

Abstract. Static and deterministic vehicle routing problems cannot be used in many real-life systems, as input data are not reliable and revealed over time. In this thesis, we study a pickup and delivery problem with time windows accounting for maximum ride time constraints – the so-called dial-a-ride problem – in its static and dynamic variant, and we make specific proposal on robust optimization models for this problem. To solve the static model, we develop a branch-and-price approach that handles ride time constraints in the process of generating feasible vehicle routes in the course of the optimization procedure. Our work is focussed on the pricing problem solver and acceleration techniques for the branch-and-price approach. Our numerical results show that the method is competitive compared to existing approaches that are based on branch-and-cut. In the dynamic context, where some input data are revealed or modified over time, we apply our branch-and-price algorithm for re-optimization in a rolling horizon approach.

Keywords: dial-a-ride problem, dynamic programming, branch-and-price, dynamic optimization, robust optimization.

Acknowledgments

I would like to express my thanks to François Vanderbeck, not only for advising me during this PhD program, but also for supporting me to move in a direction which is of my interest. Despite his busy schedule, he tried to make himself available as fast as it gets to address my questions. I am happy that I had the opportunity to work with him and to learn concepts from very elementary to the most advanced ones from him.

I am also grateful to Ruslan Sadykov for helping me in the job with BaP-Cod, as well as Boris Detienne who provided me with very useful remarks during the last couple of weeks before my PhD defend session. Many thanks to Prof. Saïd Hanafi and Dr. Antoine Jouglet for accepting to review my thesis and participating in my defend session as jury members. I also thank to Prof. Dominique Feillet and Prof. François Clautiaux for accepting to participate as a member of the jury.

I would like to thank all members of RealOpt specially Pierre Pesneau and Romain Leguay who always kindly helped me during my stay there. Thanks a lot to Iraj Mortazavi, Pauline Raout and Philippe Depouilly who helped me a lot at the beginning of my arrival. I never forget the time that Sylvain Allemande built an English keyboard for me manually; it was really nice of him. Many thanks to the secretaries of IMB as well as INRIA; special thanks to Catherine, Ida and Karine who supported me a lot during the last six months of my PhD program to get into the IMB building.

I am also very happy that my stay there allowed me to made a long lasting friendship with Grégory, Alberto, Johana, Adrien, Louiza, Amine, Antoine, Clémentine, Alex, Guillaume and Manik; they not only made my stay in Bordeaux memorable, but also supported me during the difficult times even

from far.

Finally, my great thanks is to my family especially my parents, who have always supported me, both mentally and financially, to make my dreams real despite their own worries. I would like to dedicate this work to them.

Bordeaux, July 2014, Nastaran Rahmani

Contents

Introduction

Le Dial-a-Ride (DARP) est une variante du problème de ramassage et de livraison avec intervalles de temps: les clients doivent être transportés depuis un point de départ vers une destination. Non seulement les coûts d'exploitation des véhicules doivent être réduit au minimum, mais également le confort de l'utilisateur est pris en compte par les contraintes et l'objectif. Les temps d'attente et le temps de trajet (le temps passé dans le véhicule) sont deux critères qui modélisent le confort de l'utilisateur. Les services de transport porte-à-porte pour les personnes agées et handicapées ainsi que les services de navette reliant les aéroports aux hôtels sont des applications de la DARP.

L'évolution dynamique et la fiabilité des données sont les deux dimensions importantes qui doivent être prises en compte, lorsque le problème est étudié dans un contexte réel. L'évolution de l'information se rapporte au fait que les données sont révélées au cours de la période de planification, tandis que la fiabilité de l'information traduit l'incertitude possible sur les données. La valeur exacte de données incertaines peut être déterminée dans le processus d'exécution des routes.

Dans cette thèse, le problème (DARP) est étudié à selon trois axes différents:

- le contexte **statique** et déterministe, où toutes les données sont connus à l'avance.
- le contexte **dynamique** (ou en ligne), où une partie des données qui n'est révélée que lors de la planification (manipulation de l'évolution dynamique des données).

- le contexte **robuste**, où il existe une incertitude sur les données (manipulation de la fiabilité des données).

Dans le schéma de ré-optimisation dynamique, la solution du problème statique est utilisée comme solution de base de référence. L'approche robuste pourrait être présentée comme une autre façon de définir une solution de base pour le régime de ré-optimisation.

Dans le premier chapitre de cette thèse, nous examinons les variantes actuellement étudiées dans la littérature pour le cas statique. Puis, nous présentons deux formulations mathématiques pour le problème hétérogène. Le premier modèle est la formulation naturelle où le confort de l'utilisateur est modélisé par des contraintes sur le temps passé dans le vehicule. Le modèle générique comprend également des pénalités pour les temps d'attente aux points de ramassage et aux nœuds de livraison ainsi qu'un coût appliqué à la durée du trajet. Cependant, plusieurs méthodes de résolution omettent ces coûts tout en gardant les intervalles de temps et les limitations sur la durée des trajets. Le second modèle repose sur une hypothèse restrictive simplifiée: l'hypothèse de "regroupement de l'information" (i.e regroupement de la donnée du point de ramassage et du point de livraison), où le nœud de livraison doit être signifié immédiatement après son ramassage. L'hypothèse de regroupement de l'information réduit le problème au problème de routage de véhicule bien connu avec des intervalles de temps.

Le deuxième chapitre est consacré à un problème de véhicule unique qui apparaît comme un sous-problème dans une approche de décomposition de la DARP. Nous développons des algorithmes de programmation dynamique (les méthodes "d'étiquetage avant"[1]) pour différentes variantes du problème de plus court chemin avec de ressource (RCSPP) qui se pose comme un modèle pour d'optimisation à un seul vèhicule. Nous analysons leur complexité théorique et nous les comparons expérimentalement.

Il y a de nombreux problèmes techniques qui se posent dans ces méthodes de résolution, tels que: comment gérer l'énorme espace d'état en utilisant des outils de projection (agrégation); comment choisir le bon compromis entre l'exhaustivité des tests de dominance et leur temps de calcul; et comment éviter les cycles. Sur ce dernier point, nous avons intégré les dernières technologies développées pour routage des véhicules et les avons adaptées à la DARP en incluant le probléme du plus court chemin élémentaire et celui du "NG-path". La manipulation des contraintes de temps de trajet dans le processus d'étiquetage avant est difficile. Tandis que les fenêtres temporelles conduisent la solution à une mise en service le plus tôt possible, les contraintes de temps de trajet et la minimisation du coût incite à retarder le temps de service au niveau des nœuds de ramassage. Pour gérer cette difficulté, nous décomposons la solution en deux étapes: d'abord l'itinéraire est fixée, puis la faisabilité de l'ordonnancement est vérifiée. La deuxième partie est développée sur la base du concept de relation de précédence généralisée[2]. C'est le point clé de ce travail.

Dans le troisième chapitre, nous développons une approche branch-and-price pour la version statique du DARP. L'approche de la solution repose sur une reformulation de Dantzig-Wolfe des programmes linéaires à variables mixtes entiers présentés dans le premier chapitre. Les sous-problèmes sont d'abord résolus avec la meilleure technologie développée dans le deuxième chapitre. Ensuite, certaines techniques d'accélération sont théoriquement discutées et numériquement expérimentées. L'approche globale est prouvée efficace dans le traitement des cas dont les tailles ont été considérés comme un défi dans la littérature.

[1] forward labeling
[2] generalized precedence relation

Dans le quatrième chapitre, la méthode statique est utilisée dans une approche sur horizon glissant pour traiter le cas dynamique, où au moins une partie des données est révélée en temps réel. Nous installons un système de simulation pour générer des demandes de clients aléatoires de manière dynamique. La ré-optimisation soulève des questions très spécifiques en termes de modélisation et de résolution du problème, par exemple: quelles sont les recours possible d'un véhicule, comment construire une route à partir chemin partiel existant, comment faire face à plusieurs sous-problèmes non identiques (chaque véhicule étant désormais différenciés par ses tâches en cours).

Le cinquième chapitre porte sur les perspectives sur les approches d'optimisation robustes pour le DARP. Les premiers temps de mise en service sont les paramètres d'incertitude que nous avons considérés. Nous utilisons le framework d'optimisation à une étape[3] afin de modéliser notre cas.

Enfin, la conclusion résume les contributions de notre travail.

[3]single-stage robust optimization

Introduction

The Dial-a-Ride Problem (DARP) is a variant of the pickup and delivery problem with time windows: customer have to be transported between an origin and a destination. Not only the vehicle operational costs are minimized, but also the user inconvenience is taken into account via the constraints and the objective. The waiting times and the ride time (the time spent into the vehicle) are two criteria that model the user inconvenience. The door-to-door transportation services for elderly and disabled people together with the shuttle bus service connecting airports and customer hotels are some applications of the DARP.

Dynamic evolution and reliability of the input information are two important dimensions that must be taken into account, when the dial-a-ride problem is studied in a real-world context. The evolution of information refers to the fact that data are revealed over the planning horizon, while the reliability of information reflects the possible uncertainty on input data. The true value of uncertain data can be revealed in the process of executing the routes.

In this thesis, the dial-a-ride problem is studied from three different perspectives :

- the **static** and deterministic context, where all input data are known beforehand.
- the **dynamic** (or online) context, where some of input data are revealed over the planning horizon (handling the dynamic evolution of data).
- the **robust** context, where there is uncertainty on the input data (handling the reliability of data).

In the dynamic re-optimization scheme, the static problem solution is used as a base line reference solution. The robust approach could come as an alternative way to define a base line solution for the re-optimization scheme.

In the first chapter of this thesis, we review the problem variants currently studied in the literature for the static case. Then, we introduce two mathematical formulations for the heterogenous dial-a-ride problem. The first model is the natural formulation for heterogenous dial-a-ride problem where the user inconvenience (which is a specific characteristic of the dial-a-ride problem) is modeled through ride time constraints. The generic model also includes penalties for waiting times at pickup and delivery nodes and a cost applied to ride times, but several solution methods omit these costs while keeping the time windows and bounds on ride times. The second model relies on a simplifying restrictive assumption: the "request consolidation" assumption, where the delivery node must be served immediately after its pickup. The request consolidation assumption reduces the problem to the well-known vehicle routing problem with time windows.

The second chapter is devoted to one-vehicle problem that appears as a subproblem in a decomposition approach to the DARP. We develop different dynamic programming algorithms (forward labeling approaches) for different variants of the underlying resource constraint shortest path problem (RC-SPP) that arises as a model for the price collecting one-vehicle problem. We analyze their theoretical complexity and compare them experimentally.

There are numerous technical issues arising in these solution methods, such as: how to manage the huge state space using projection tools (aggregation); how to stricke the right compromise between completeness of the dominance tests and their computing time; and how to avoid cycles. On the latter issue, we imported the latest technologies developed for the vehicle routing and adapted them to the DARP: enforcing path elementarity versus using NG-path. Handling ride time constraints in the process of forward labeling approach is challenging. While time windows drives the solution to setting service times as early as possible, the ride time constraints and minimization bring an incentive to delaying service time at pickup nodes. To manage this difficulty, we decompose the solution in two steps: first the routing solution is fixed, then the existence of a time feasible schedule on

the route is verified. The second part is developed based on the concept of generalized precedence relations. It is key point in this work.

In the third chapter, we develop a branch-and-price approach for the static DARP. The solution approach relies on a Dantzig-Wolfe reformulation of the mixed-integer linear programs presented in the first chapter. The subproblems are first solved with the best technology developed in the second chapter. Then, some accelerating techniques are theoretically discussed and numerically experimented. The overall approach are proved efficient in dealing with instances of the size that have been considered as challenging in the literature.

In the fourth chapter, the static method is used within a rolling time horizon approach to address the online case, where at least part of the input data is revealed in real time. We setup a simulation scheme to generate random customer requests online. The re-optimization implies very specific issues in terms of modeling and solving the problem, such as: what are the allowed recourse action of the vehicle, how to build a route from an existing partial path, how to deal with many non-identical subproblems (as each vehicle is now differentiated by its ongoing duties).

The fifth chapter is about perspectives on robust optimization approaches for the DARP. The earliest start service times are the parameter uncertainty that we have considered. We use the single-stage robust optimization framework to model our case.

Finally, the conclusion summarizes the contributions of our work.

1

The Dial-a-Ride Problem

Introduction

The Dial-a-Ride Problem (DARP) is a variant of the pickup and delivery problem with time windows, where people have to be transported between an origin and a destination. In the transportation of people, not only the operational costs are minimized, but also the user inconvenience. The door-to-door transportation services for elderly and disabled people together with the shuttle bus service connecting airports and customer hotels are some applications of DARP. In this chapter, first we review the problem variants currently studied in the literature, then we introduce two mathematical formulations for the heterogenous dial-a-ride problem. The second model relies on a simplifying restrictive assumption: the "request consolidation" assumption, where the delivery node must be served immediately after its pickup. Such assumption reduces the problem to the well-known vehicle routing problem with time windows.

1.1. Review on Problem Variants

The dial-a-ride problem is to cover a set of transportation requests with a fleet of vehicles. Each request is defined by an origin (pickup node) and a destination (delivery node). There exist different variations of the problem. In some cases, the challenge is to first determine a fleet size and a compo-

sition capable of satisfying all requests, while for other variations, a fixed fleet size is given and the goal is to maximize the number of requests that can be served. Serving some of the requests using available vehicle fleet and applying extra vehicles (like taxis) can be a compromise of the two variations.

In 2007, Cordeau and Laporte [13] provided an excellent review on the dial-a-ride problem variants. Among 25 reviewed papers, there exist only 5 contributions on exact approaches, while the rest were using heuristic approaches. In the domain of exact approaches the work of Cordeau [12] is mostly cited. He proposed a three-index formulation and solved instances with up to four vehicles and 32 requests using a branch-and-cut approach. In his work, it is mentioned that handling the ride time constraint is far from trivial in a column generation context. Later Robke et al. [39] applied a branch-and-cut approach on a two-index formulation. Their work resulted in solving instances with up to eight vehicles and 96 requests. Although the latter formulation is better, it only works with a homogeneous fleet of vehicles.

In 2011, Parragh et al. [32] studied dial-a-ride problems with heterogenous users and fleet. They successfully adapted the state-of-the-art branch-and-cut algorithm proposed by Cordeau [12]. In 2012, the same author studied a more complex heterogeneous dial-a-ride problem, where two types of vehicles and four different transportation modes were considered together with driver related constraints [31]. They proposed a three-index and a set partitioning formulation for their problem. Finally, the linear programming relaxation of the set partitioning formulation is solved using a column generation approach. They also put their proposed variable neighborhood search heuristic into a collaborative framework with the column generation algorithm. To manage the difficulty of the ride time constraint in dynamic programming, Parragh et al. [31] considered the maximum ride time implicitly in terms of time windows both at the pickup and the delivery node.

In this thesis we are studying the heterogeneous dial-a-ride problem, where the fleet size is given. Different variations of the problem are assessed under assumptions outlined in next section.

1.2. Input Data and Notations

The dial-a-ride problem is mathematically defined on a complete symmetric directed graph $\bar{G} = (\bar{N}, \bar{A})$, where $\bar{N}$ is the set of actual sites and each $(i, j) \in \bar{A}$ represents shortest path between $i \in \bar{N}$ and $j \in \bar{N}$. Requests are defined over graph $\bar{G}$ by their origin (pickup node) and destination (delivery node). Let $\mathcal{R} = \{0, 1, ..., |\mathcal{R}|\}$ denote the set of requests. Each request $r \in \mathcal{R}$ entails a demand of a certain load for a transportation service from a pickup node to a delivery node within predefined time windows. Furthermore, for each pickup node (and similarly each delivery node) a service duration is defined. Thus, each request $r \in \mathcal{R}$ is characterized by:

- $c_r^f \in \mathbb{R}^+$, the flow time cost of the request.
- $b_r^f \in \mathbb{R}^+$, the flow time bound of the request.
- $p_r \in \bar{N}$, the pickup node of the request.
- $d_r \in \bar{N}$, the delivery node of the request.
- $q_r \in \mathbb{Z}^+$, the load of the request.
- $e_{p_r} \in \mathbb{R}^+$, the earliest time for starting service at the pickup node of the request.
- $l_{p_r} \in \mathbb{R}^+$, the due date for the start of the service at the pickup node; it can be violated at a cost.
- $\bar{l}_{p_r} \in \mathbb{R}^+$, the deadline for starting service at the pickup node; it cannot be violated.
- $e_{d_r} \in \mathbb{R}^+$, as the earliest time for delivering (i.e., for starting the delivery service) at the delivery node.
- $l_{d_r} \in \mathbb{R}^+$, the due date for starting the delivery service at the delivery node.
- $\bar{l}_{d_r} \in \mathbb{R}^+$, the deadline for starting the delivery service at the delivery node.
- $s_{p_r} \in \mathbb{R}^+$, the service duration for the pickup node.
- $s_{d_r} \in \mathbb{R}^+$, the service duration for the delivery node.

Let $P = \{p_r : p_r \text{ is a pickup node of request } r \in \mathcal{R}\} \subseteq \bar{N}$ be the set of pickup nodes and $D = \{d_r : d_r \text{ is a delivery node of request } r \in \mathcal{R}\} \subseteq \bar{N}$ be the set of delivery nodes. Assume that K_i is the set of available vehicles of type $i \in \{0, ..., m-1\}$. Hence, $K = \cup_{i=0}^{m-1} K_i$ is the set of all available vehicles. Initially, all the vehicles are located in the depot. Each vehicle $k \in K_i$ has a capacity Q_i. Assume that $Q = \max_i Q_i$. Hence, for each vehicle $k \in K$, we define $Q_k \in \mathbb{Z}^+$, as the capacity of the vehicle.

Let us now consider a virtual network $G = (N, A)$ that will be useful to introduce the mathematical formulation of the problem. Define $P' \subseteq P \times \mathcal{R}$, $D' \subseteq D \times \mathcal{R}$ and O' as follows:

$$P' = \{(p_r, r) : r \in \mathcal{R}\} \quad (1.1)$$
$$D' = \{(d_r, r) : r \in \mathcal{R}\} \quad (1.2)$$
$$O' = \{o^+, o^-\} \quad (1.3)$$

For simplicity, we use p_r to represent $(p_r, r) \in P'$ and d_r to represent $(d_r, r) \in D'$. Finally, o^+ and o^- are the duplication of the depot node that represent the depot on departure and the return to the depot node, respectively.

Then, let $N = P' \cup D' \cup O'$. Note that $P' \cap D' = \emptyset$, since the elements of P' and D' are distinguished by the request index r and for each request the pickup node and the delivery node cannot be equal.

All in all, with each node $i \in N$ is associated:

- a load q_i.
- a nonnegative service duration s_i.
- a time window $[e_i, \bar{l}_i]$, where e_i and $\bar{l}_i$ represent the earliest time and the latest possible time at which service may begin at node i.
- a due date, $l_i \in [e_i, \bar{l}_i]$, which can be violated at a cost.
- a waiting cost, c_i^w, that penalizes the violation of the due date.

If $i \in O'$, then $q_i = 0$ and $s_i = 0$. If $i \in P'$, $q_i > 0$; if $i \in D'$, $q_i < 0$. It is clear that for each $r \in \mathcal{R}$, $|q_{p_r}| = |q_{d_r}|$.

1.3. Model Assumptions

We are interested to define different assumptions that can be imposed to the problem.

Assumption 1 (Load Splitting) *This assumption specifies different alternatives to cover the load of a request:*

a. The load of a request cannot be split.

b. The load of any request may be split.

In the rest of this thesis, option (a) of Assumption 1 is considered by default. In the case where option (b) of any assumptions is used, it will be indicated.

Assumption 2 (Ride Time Value) *This assumption defines different alternatives to calculate the ride time:*

a. The ride time is counted from the pickup time (= actual ride time).

b. The ride time is counted from the earliest pickup time (= waiting time + ride time).

In the sequel, option (a) of Assumption 2 is considered by default.

Assumption 3 (Request Interfacing) *This assumption defines different alternatives to sequence the pickup node and the delivery node of each request:*

a. A vehicle can carry several requests at a time by doing a sequence of pickups before their respective delivery.

b. If a vehicle serves the pickup node of request $r \in \mathcal{R}$, then it must immediately serve its delivery node before doing any further pickup.

c. A vehicle can carry up to n different requests at the time.

d. A vehicle must return to the empty state after at most n deliveries.

In the sequel, option (a) of Assumption 3 that imposes no restriction on request interfacing is considered by default. Option (b) is the so-called **"request consolidation"** assumption that simplifies greatly the dial-a-ride problem by reducing it to a standard Vehicle Routing Problem with Time Windows (VRPTW). Option (c) limits the number of "open" requests; with $n = |\mathcal{R}|$, it reproduces option (a). Option (d) limits the length of a sequence of interfacing pickups and deliveries for different requests; with $n = |\mathcal{R}|$, it reproduces option (a).

Assumption 4 (Route Duration) *This assumption defines a driver related constraint:*

a. *A maximum value is not considered for the route duration.*

b. *A maximum value, T_k, is considered for the route duration.*

b. *A maximum value, T_k, is considered for the duration of a sequence request interfacing (i.e., the duration during which the vehicle is not empty).*

In the sequel, option (a) of Assumption 4 is considered by default.

Assumption 5 (Timing Costs) *This assumption defines the impact of waiting time and ride time in the objective:*

a. *Waiting, lateness and ride time have positive cost in the objective.*

b. *Only the waiting time is penalized with a cost in the objective, while ride times are bounded as a constraint of the system.*

c. *Both waiting times and ride times are bounded as constraints of the system but none of them appear in the objective function.*

In the sequel, option (a) of Assumption 5 is considered by default in the models, but most solution approaches are developed under option (b) or (c) of Assumption 5.

Assumption 6 (Ride Time Constraint) *This assumption defines the impact of ride time constraints on the feasibility set:*

a. *Some ride time constraints are tight (have an impact in the definition of feasible solutions).*

b. *Ride time constraints can safely be ignored, as once time window constraints are enforced, ride time constraints are naturally satisfied.*

Some of the algorithms that we develop later are exact under Assumption 6-b but only heuristic under Assumption 6-a;

1.4. Mathematical Formulation

In this section two different mathematical formulations are presented for the heterogeneous dial-a-ride problem. The first formulation is the compact formulation under option (a) of all assumptions, that is load splitting is not allowed, the actual ride time is calculated, the sequencing of pickup and delivery nodes does not follow any specific assumptions, for the route duration we do not consider any upper bound, and all timing cost are considered. This formulation is an extension of the one introduced by Cordeau [12] for the homogenous DARP, while they assumed the maximum route duration. In section 1.4.2 another compact formulation is proposed using option (b) of the third assumption. Request consolidation allows us not to consider the ride time constraint, the precedence constraint, and the capacity constraint.

1.4.1 Generic Compact Formulation

To model the problem as a mixed integer linear program, first we define the decision variables as follows:

- $x_{ij}^k \in \{0,1\}$ is 1 if vehicle $k \in K$ travels on arc $(i,j) \in A$ and 0 otherwise ($i \neq j$, $i \neq o^-$, $j \neq o^+$, and $i, j \in N$).
- $t_i^k \in [e_i, \bar{l}_i]$ is the start service time of vehicle k at node $i \in N$.[1]
- $Q_i^k \in [0, Q]$ is the load of the vehicle k after visiting node $i \in N$.

[1] It is possible to assume $e_{o^+} = 0$ and therefore $t_{o^+}^k = 0$ for all $k \in K$.

- $w_i^k \in [0, \bar{l}_i - l_i]$ is the amount of time that customer $i \in N \setminus \{o^+, o^-\}$ waits for vehicle k because it arrives after due date l_i.
- $f_r^k \in [0, b_r^f]$ is the ride time for request $r \in \mathcal{R}$ on vehicle $k \in K$. Under assumption (2.a) , $f_r^k = t_{d_r}^k - t_{p_r}^k - s_{p_r}$. Under assumption (2.b), $f_r^k = t_{d_r}^k - e_{p_r}$.
- $y_r^k \in \{0, 1\}$ is 1 if vehicle k is taking care of request $r \in \mathcal{R}$ (with its full load under Assumption 1.a) and 0 otherwise.

The formulation is then:

$$\min \sum_{k\in K}\{\sum_{i\in N}\sum_{j\in N} c_{ij}^k x_{ij}^k \;+\; \alpha \sum_{i\in N\setminus O'} c_i^w w_i^k + \beta \sum_{r\in\mathcal{R}} c_r^f f_r^k\} \tag{1.4}$$

$$\sum_{k\in K} y_r^k = 1 \quad \forall r \in \mathcal{R} \tag{1.5}$$

$\forall k \in K$:

$$y_r^k - \sum_{j\in N\setminus\{o^+\}} x_{p_r j}^k = 0 \quad \forall r \in \mathcal{R} \tag{1.6}$$

$$y_r^k - \sum_{i\in N\setminus\{o^-\}} x_{i d_r}^k = 0 \quad \forall r \in \mathcal{R} \tag{1.7}$$

$$\sum_{j\in P'} x_{o^+ j}^k = 1 \tag{1.8}$$

$$\sum_{j\in N\setminus\{o^-\}} x_{ji}^k - \sum_{j\in N\setminus\{o^+\}} x_{ij}^k = 0 \quad \forall i \in N\setminus O' \tag{1.9}$$

$$\sum_{i\in D'} x_{i o^-}^k = 1 \tag{1.10}$$

$$t_i^k + s_i + t_{ij} - M_{ij}^k(1-x_{ij}^k) \le t_j^k \quad \forall i \in N\setminus\{o^-\}, j \in N\setminus\{o^+\} \tag{1.11}$$

$$Q_i^k + q_j - W_{ij}^k(1-x_{ij}^k) \le Q_j^k \quad \forall i \in N\setminus\{o^-\}, j \in N\setminus\{o^+\} \tag{1.12}$$

$$e_{p_r}\, y_r^k \le t_{p_r}^k \le \bar{l}_{p_r}\, y_r^k \quad \forall r \in \mathcal{R} \tag{1.13}$$

$$e_{d_r}\, y_r^k \le t_{d_r}^k \le \bar{l}_{d_r}\, y_r^k \quad \forall r \in \mathcal{R} \tag{1.14}$$

$$w_i^k \ge t_i^k - l_i \quad \forall i \in N\setminus O' \tag{1.15}$$

$$0 \le w_i^k \le \bar{l}_i - l_i \quad \forall i \in N\setminus O' \tag{1.16}$$

$$q_r\, y_r^k \le Q_{p_r}^k \le Q_k\, y_r^k \quad \forall r \in \mathcal{R} \tag{1.17}$$

$$0 \le Q_{d_r}^k \le (Q_k - q_{d_r})\, y_r^k \quad \forall r \in \mathcal{R} \tag{1.18}$$

$$t_{d_r}^k - t_{p_r}^k - s_{p_r} \le f_r^k \quad \forall r \in \mathcal{R} \tag{1.19}$$

$$t_{p_r d_r}\, y_r^k \le f_r^k \le b_r^f\, y_r^k \quad \forall r \in \mathcal{R} \tag{1.20}$$

The objective function (1.4) is to minimize the routing cost and the timing costs. Parameters α and β can be tuned to balance these costs. Constraint (1.5) is the assignment constraint, which ensures that each request is covered exactly once. Constraints (1.6) and (1.7) impose that if a request is handled by a vehicle, both pickup and delivery must be done in the route. Constraints (1.8) to (1.10) guarantee that the route of each vehicle k starts at

the source node o^+ and ends at the sink node o^-. The feasibility of time and load variables for arc (i, j) is checked by constraints (1.11) and (1.12). Note that in these constraints M_{ij}^k and W_{ij}^k are the "big-M" constants which are defined in Remark 1. Besides, t_{ij} in constraint (1.11) represents the distance travel between node i and j. The time window are imposed in constraints (1.13) - (1.14). The tardiness of the vehicle at each node can be measured by constraints (1.15) and (1.16). Constraints (1.17) and (1.18) formulate the capacity constraints. The ride time of each request is defined and bounded through constraints (1.19) - (1.20).

Remark 1 *For the "big-M" constant, one can set* $W_{ij}^k = \min\{Q_k, Q_k + q_j\}$ *and* $M_{ij}^k = \bar{l}_i + s_i + t_{ij}$ *for each* $i, j \in N$ *and* $k \in K$.

Remark 2 *Under assumption (2.b) constraints (1.19) and (1.20) will be replaced by the followings*

$$t_{d_r}^k - e_{p_r} \leq f_r^k \quad \forall r \in \mathcal{R}, k \in K \tag{1.21}$$

$$t_{p_r d_r}\, y_r^k \leq f_r^k \leq b_r^f\, y_r^k \quad \forall r \in \mathcal{R}, k \in K \tag{1.22}$$

1.4.2 Compact Formulation under the Consolidation Assumption

Making Assumption 3-b introduced in section 1.2, the set N of the virtual graph G will reduce to $N = \mathcal{R} \cup O'$ and $A = \{(i, j) : i \in N, j \in N, i \neq j\}$. In this case, the decision variables are reduced to

- $x_{ij}^k \in \{0, 1\}$ is 1 if and only if vehicle $k \in K$ travels on arc $(i, j) \in A$ and 0 otherwise ($i \neq j$, $i \neq o^-$, $j \neq o^+$, and $i, j \in N$).
- $t_i^k \in [e_{p_i}, \bar{l}_{p_i}]$ is the the start service time of vehicle k at node $i \in \mathcal{R}$.[2]
- $Q_i^k \in [0, Q]$ is the load of the vehicle k after visiting node $i \in N$.
- $y_i^k \in \{0, 1\}$ is 1 if vehicle k is taking care of request $i \in \mathcal{R}$ (with its full load under Assumption 1.a) and 0 otherwise.

[2] $t_{o^+}^k = 0, t_{o^-}^k \in [0, T]$ for all $k \in K$; where T is the length of the planning horizon.

Then the formulation takes the following form, in which the objective is to minimize the routing cost:

$$\min \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ij}^k x_{ij}^k \tag{1.23}$$

$$\sum_{k\in K}\sum_{j\in\mathcal{R}\cup\{o^-\}} x_{ij}^k = 1 \quad \forall i \in \mathcal{R} \tag{1.24}$$

$\forall k \in K$:

$$\sum_{j\in\mathcal{R}\cup\{o^-\}} x_{o^+\,j}^k = 1 \tag{1.25}$$

$$\sum_{j\in N\setminus\{o^-,i\}} x_{ji}^k - \sum_{j\in N\setminus\{o^+,i\}} x_{ij}^k = 0 \quad \forall i \in \mathcal{R} \tag{1.26}$$

$$\sum_{i\in\mathcal{R}\cup\{o^+\}} x_{i\,o^-}^k = 1 \tag{1.27}$$

$$t_{o^+}^k + t_{o^+\,p_j} - M(1 - x_{o^+\,j}^k) \leq t_j^k \quad \forall j \in \mathcal{R} \tag{1.28}$$

$$t_i^k + s_i + t_{d_i\,p_j} - M(1 - x_{ij}^k) \leq t_j^k \quad \forall i \neq o^+, j \neq o^- \in \mathcal{R} \tag{1.29}$$

$$t_i^k + s_i + t_{d_i\,o^-} - M(1 - x_{i\,o^-}^k) \leq t_{o^-}^k \quad \forall i \in \mathcal{R} \tag{1.30}$$

$$e_{p_i}\, y_i^k \leq t_i^k \leq \bar{l}_{p_i}\, y_i^k \quad \forall i \in \mathcal{R} \tag{1.31}$$

$$y_i^k = \sum_{j\in\mathcal{R}\cup\{o^-\}} x_{ij}^k \quad \forall i \in \mathcal{R} \tag{1.32}$$

$$0 \leq Q_i^k \leq Q_k\, y_i^k \quad \forall i \in N \setminus \{o^+, o^-\} \tag{1.33}$$

where $t_{o^+\,p_j}$ is the distance travel between o^+ and the pickup node of request $j \in \mathcal{R}$; $t_{d_i\,o^-}$ is the distance travel between the delivery node of request $i \in \mathcal{R}$ and o^-. Finally, $t_{d_i\,p_j}$ indicates the distance travel between the delivery node of request $i \in \mathcal{R}$ and the pickup node of request $j \in \mathcal{R}$, while node i precede node j.

Assumption 7 *We further assume that* $e_{d_r} = e_{p_r} + s_{p_r} + t_{p_r\,d_r}$ *and* $\bar{l}_{d_r} = e_{d_r} + \Delta^r$*, where* Δ^r *is the length of the time window for request* r*.*

Having Assumption 7, the problem reduces to the vehicle routing problem with time windows.

Remark 3 *Note that, in this formulation, for each $r \in \mathcal{R}$ the service duration is defined as follows:*

$$s_r = s_{p_r} + t_{p_r d_r} + s_{d_r} \tag{1.34}$$

Besides, let $M = \bar{l}_{p_r} + s_r + t_{d_r p_r}$.

Conclusion

The Dial-a-Ride problem can be formulated in a compact way (using a polynomial size formulation). The presence of big-M in these formulations is an indication of the poor quality of the Linear Programming (LP) dual bound that can be expected. The block diagonal structure of the constraint matrix points to decomposition approaches. Restricting the problem by making the consolidation assumption largely simplifies the problem, reducing it to a standard VRPTW. Selecting option (b) of Assumption 5, i.e. assuming no cost on flow time, is making a large difference in the complexity of the underlying timing problem as we shall see.

2

The One Vehicle Subproblem

Introduction

Shortest path problems arise in wide variety of practical problems, both as stand-alone models and as subproblems in more complex problem settings. The vehicle routing application is an instance of such complex problem setting where the problem can be decomposed into a subproblem for each vehicle. The subproblem is a resource constrained shortest path problem. For the dial-a-ride problem the subproblem is more complex as it requires deciding on not only the sequence of visit, but also the time of visit. Note that the issue of timing is also present in vehicle routing problems with time windows, but there the timing decisions are trivial: one can simply visit the node as early as they can feasibly be visited, given the sequence of visits. While in dial-a-ride applications, the ride time constraints may induce an incentive for delaying the time of pickup.

The resource constrained shortest path problem can be solved using a dynamic programming approach. In this chapter, we review different variants of resource constraint shortest path problem, along with relevant forward labeling solution approaches.

2.1. The Shortest Path Problem

Let $G = (N, A)$ be a directed graph, where an arc cost c_{ij} is associated with each arc $(i, j) \in A$. Assume that n and m define the number of nodes and the number of arcs in the graph, respectively. Let o^+ and o^- be the source node and the sink node, respectively. The shortest path problem (SPP) is to determine a path from o^+ to o^- with minimum cost. Its linear programming formulation is as follows

$$\min \quad \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{2.1}$$

$$\text{s.t}$$

$$\sum_{j\in N\setminus\{o^+,o^-\}} x_{o^+j} = 1 \tag{2.2}$$

$$\sum_{j\in N\setminus\{o^+\}} x_{ij} - \sum_{j\in N\setminus\{o^-\}} x_{ji} = 0 \quad \forall i \in N \setminus \{o^+, o^-\} \tag{2.3}$$

$$\sum_{i\in N\setminus\{o^+,o^-\}} x_{io^-} = 1 \tag{2.4}$$

$$x_{ij} \geq 0 \quad \forall (i, j) \in A \tag{2.5}$$

where x_{ij} is the flow variable. Note that the above formulation models a shortest walk problem (in a walk the same node can be visited several times). In the sequel, we assume the shortest walk problem when we speak of shortest path. However, we shall sometime enforce that no more than one visit to each node can take place. Then, we shall speak of an elementary path.

It is possible to define and impose different assumptions to the problem which will result in different types of the problem:

1. Single Source Shortest Path Problem, where the aim is to find shortest path from one node to all other nodes when arc costs are non-negative.

2. Single Source Shortest Path Problem with arbitrary arc cost.

3. All-Pairs Shortest Path Problem, where shortest paths from every node to every other node must be found.

To solve different variants of shortest path problem, the network flow literature typically classifies algorithmic approaches into two groups: label setting and label correcting. At each step, both approaches iteratively assign tentative distance labels to nodes; the distance labels contain an upper bound on the shortest path distances; a label becomes *permanent* when its value represent the true shortest path distance. The existing approaches vary in how they update the distance labels gradually and how they converge toward the shortest path distances. Label setting-algorithms designate one label as permanent at each iteration. However, label correcting algorithms consider all labels as temporary until the last step when they all become permanent.

Although label-setting algorithms have much better worst-case complexity bounds, they can only treat shortest path problems with a special structure: either a special network topology (acyclic graphs with arbitrary arc costs) or a special cost structure (non-negative arc costs). The problem is more complex, when networks have arbitrary costs and arbitrary topology. In terms of computational complexity, the elementary shortest path problem is NP-complete unless the graph contains no negative cost cycles. Several types of modifications to the various label-correcting algorithms can detect the existence of negative cycles, if negative cycles are allowed in the network. Label correcting algorithms are more general and can be applied to all classes of problems (including those with negative arc costs, which are detected in the process) [2].

Following the goal of this chapter, let us review the forward labeling approach for different variants of shortest path problem. The procedure recursively builds a shortest path solution from dominant partial solutions. A partial solution is defined by a label that records the current state of the solution building, recording only the information required to pursue the construction of the solution. Let us first define the possible states and associated labels. For the standard shortest path problem, **a label** L contains the following features:

- $\mathrm{i}^L \in N$ is the end-node of the partial path defined by the label.
- c^L is the accumulated cost until that node.
- p^L is a pointer to the predecessor label of L that represent the partial

path that was extended to obtain L.

Let $\mathcal{L}$ denotes the list of labels to be processed. The forward labeling approach for the standard shortest path problem, initializes $\mathcal{L}$ with a single label representing a null path that ends at the source node with zero cost. Besides, $\mathcal{L}$(i) denotes the set of labels defined with end node i. As shown in Table 2.1, at each iteration of the algorithm, a label $L \in \mathcal{L}$ is chosen and the process is to extend this label to any other node in its adjacency list $A(\mathrm{i}^L)$. In the shortest walk problem, a label L can be extended to any other node in the network. Then, the newly created label, L', will be added to $\mathcal{L}$ if it is not dominated by an already existing label in $\mathcal{L}(\mathrm{i}^{L'})$. Label L' is linked to L by a backward pointer $\mathrm{p}^{L'}$. The algorithm stops when $\mathcal{L}$ is empty. Finally, a label with minimum cost at sink node returns an optimal solution. The worst-case computational complexity of the algorithm is presented in the following observation. Before outlining the observation, let $C = \max\{c_{ij} : (i,j) \in A\}$.

Observation 2.1.1 *If graph G does not include any negative cost cycles, then the worst-case computational complexity of the algorithm of Table 2.1 is $O(nmC)$.*

Proof To analyze the worst case complexity observe that line 2 of the algorithm takes $O(1)$ operations, which will be repeated n times. Line 3 takes $O(1)$ operations. For line 4, we need to count the number of pops from $\mathcal{L}$. This number is bounded by the number of pushes done at line 10. A push can only happen when there is a strict cost improvement for a given node (see line 6). Note that C is the maximum possible value of arc costs and all data are integer. As a path contains at most $n-1$ arcs, one can have at most $O(nC)$ one unit of cost improvement at each iteration for a given node, and hence one can have $O(nC)$ pushes at the given node. When a node $i \in N$ is pushed, this node will be selected in a later iteration of the algorithm, and line 5 scans the adjacency list of i, $A(i)$. Therefore, for each node i line 6 to 10 with $O(1)$ operations will be executed $O(nC|A(i)|)$ times. Considering all the nodes of the graph, line 6 to 10 take $O(nmC) = \sum_{i \in N} O(nC|A(i)|)$ operations. This latter block of instructions dominates the time required to execute lines 1 to 3. Thus, the computational complexity of the total algorithm is $O(nmC)$.□

Table 2.1: Forward Labeling Approach for SPP

1	**for each** $i \in N$ **do**
2	$\mathcal{L}(i) \leftarrow NULL$; ($\mathcal{L}(i)$ represents the label assigned to node i.)
3	Create an initial label at o^+ with zero cost and put a copy in $\mathcal{L}(o^+)$ and $\mathcal{L}$;
4	**while** $\mathcal{L}$ is not empty, pop the first label L in $\mathcal{L}$ **do**
5	**for each** $j \in A(\mathrm{i}^L)$ **do**
6	**if** $\mathrm{c}^L + c_{\mathrm{i}^L j} < \mathrm{c}^{\mathcal{L}(j)}$ **do**
7	$L'(\text{ node } j,\ \text{cost } \mathrm{c}^L + c_{\mathrm{i}^L j},\ \text{pred } L)$;
8	$\mathcal{L} \setminus = \mathcal{L}(\mathrm{i}^{L'})$;
9	$\mathcal{L}(\mathrm{i}^{L'}) \leftarrow L'$;
10	$\mathcal{L}.pushback(L')$;
11	**return** $\mathcal{L}(o^-)$;

Observation 2.1.2 *Assume that G is an acyclic graph and the nodes are handled in topological order (i.e. a vector indexed in topological order of the network nodes is used to implement $\mathcal{L}$). Then, the worst-case computational complexity of the proposed algorithm is $O(m)$.*

Proof The proof is similar to the one presented for observation 2.1.1. Note that in this case $\mathcal{L}$ is a vector of labels whose indexing is in the topological order of the network nodes. When a label is poped at a given node (line 4), that node does not belong to the adjacency list of the remaining nodes in $\mathcal{L}$. Hence, no new label associated to that node will arise in the vector. Thus, a pop can happen only once for a given node. After that, all the nodes in the adjacency list of the poped node are scanned (line 5). Considering all the nodes of the network, line 6 to 10 of the algorithm needs $\sum_{i \in N} |A(i)| = O(m)$ operations, which indicates the computational complexity of the whole algorithm. □

2.2. The Shortest Path Problem with Time Windows

The shortest path problem with time windows (SPPTW) is a generalization of the classical shortest path problem, where for each node $i \in N$ a time window $[e_i, \bar{l}_i]$ is defined. In this problem the aim is to find the least cost path between a source node o^+ and a sink node o^-, while the time window

condition at each visited node is respected. The mathematical formulation for the shortest walk problem with time windows is then as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.6}$$

s.t

$$\sum_{j\in N\setminus\{o^+,o^-\}} x_{o^+j} = 1 \tag{2.7}$$

$$\sum_{j\in N\setminus\{o^+\}} x_{ij} - \sum_{j\in N\setminus\{o^-\}} x_{ji} = 0 \quad \forall i \in N \setminus \{o^+, o^-\} \tag{2.8}$$

$$\sum_{i\in N\setminus\{o^+,o^-\}} x_{io^-} = 1 \tag{2.9}$$

$$t_i + t_{ij} - M(1 - x_{ij}) \leq t_j \quad \forall i \in N \setminus \{o^-\}, j \in N \setminus \{o^+\} \tag{2.10}$$

$$y_i \leq \sum_{j\in N\setminus\{o^+\}} x_{ij} \quad \forall i \in N \tag{2.11}$$

$$e_i\, y_i \leq t_i \leq \bar{l}_i\, y_i \tag{2.12}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{2.13}$$

$$y_i \in \{0,1\} \quad \forall i \in N \tag{2.14}$$

where x_{ij} variables represent flow in the network, t_i measures at what time the path enters node i, and y_i is a binary variable; it takes value 1 if node i is visited and 0 otherwise. The travel time between i and j is given by t_{ij} as an input data, and M is a big number. Note that although a node can be visited several times, the time of service is defined only once. The time window constraints will naturally limit the cycles in the walk if they are tight.

This problem was first introduced by Desrosiers et al. [15] and Desrosiers et al. [16] as a subproblem for the multiple traveling salesman problem with time windows. In 1988, Desrochers and Soumis [14] proposed *Generalized Permanent Labeling Algorithm* (GPLA), which is a generalization of the best-first strategy of Dijkstra's algorithm for the shortest path problem; GPLA can solve the problems with up to 2500 nodes and 250,000 arcs in a few minutes, which outperforms the methodology presented in [15]. Later in 1998, Powell and Chen [34] proposed an adaptation of the threshold algorithm, which is substantially faster than the label-setting algorithm presented in [14].

An extension of the forward labeling algorithm presented in previous section is obtained by extending the features of a label L as follows:

- $\mathrm{i}^L \in N$ is the end-node of the partial path defined by the label.
- c^L is the accumulated cost upto that node.
- t^L is the time at which the last node is visited.
- p^L is a pointer to the predecessor label of L.

Let $\mathcal{L}$ be a list of active labels that remain to be extended to any other node and $\mathcal{L}(i) \subseteq \mathcal{L}$ be a list of non-dominated labels at node $i \in N$. List $\mathcal{L}$ is initialized with a single label representing a null path that ends at the depot with zero cost and no consumption of resources. An arc extension from label L to node $j \in A(\mathrm{i}^L)$ is considered by creating a new label L' at node j as follows:

$$\mathrm{i}^{L'} = j \tag{2.15}$$
$$\mathrm{c}^{L'} = \mathrm{c}^L + c_{\mathrm{i}^L j} \tag{2.16}$$
$$\mathrm{t}^{L'} = \max\{e_j, \mathrm{t}^L + t_{\mathrm{i}^L j}\} \tag{2.17}$$
$$\mathrm{p}^{L'} = L \tag{2.18}$$

This new label must satisfy the feasibility tests:

$$\mathrm{t}^{L'} \leq \bar{l}_j \ , \tag{2.19}$$
$$\mathrm{t}^{L'} + t_{jo^-} \leq \bar{l}_{o^-} \ . \tag{2.20}$$

i.e. one checks if going to j allows to obey the time window and also allows time to return to the depot. Not satisfying condition (2.20) causes an early termination of the partial path at L'; in this way the generation of useless labels is avoided.

In the dynamic programming context, when a new label L' is created, it is interesting to verify if L' **dominates** some labels in $\mathcal{L}$ and if some already created labels in $\mathcal{L}$ dominate this new label. Label L' dominates label L''

$(L' \preceq L'')$ if

$$\mathrm{i}^{L'} = \mathrm{i}^{L''} \tag{2.21}$$
$$\mathrm{t}^{L'} \leq \mathrm{t}^{L''} \tag{2.22}$$
$$\mathrm{c}^{L'} \leq \mathrm{c}^{L''} \tag{2.23}$$

Strict dominance rule can happen if at least one of the above inequalities is strict, which will be noted as $L' \prec L''$.

Note that a partial dominance test can be obtained via applying the dominance rule in one direction only (either $L' \prec L''$ or $L'' \preceq L'$). On the other hand, dominance rule can be checked over a subset of eligible labels. For SPPTW, one way to implement the latter idea is to check dominance only for labels with the same end-node and the same time (resource) consumption. This amounts to defining a **partial dominance test** as:

$$\mathrm{i}^{L'} = \mathrm{i}^{L''} \tag{2.24}$$
$$\mathrm{t}^{L'} = \mathrm{t}^{L''} \tag{2.25}$$
$$\mathrm{c}^{L'} \leq \mathrm{c}^{L''} \tag{2.26}$$

Tables 2.2 and 2.3 present two versions of label correcting algorithms for the SPPTW. That of Table 2.2 uses dominance rule (2.21) to (2.23) in two directions over all eligible labels, while in that of Table 2.3, the dominance rule is applied in one direction only over a subset of eligible labels using criteria (2.24) to (2.26).

To discuss the worst-case computational complexity of the proposed algorithms, let $T = \bar{l}_{o^-} - e_{o^+}$ be the length of the planning horizon and $C = \max\{c_{ij} : (i,j) \in A\}$.

Observation 2.2.1 *If graph G does not contain any negative cost cycle, then the worst-case computational complexity of the algorithm proposed in Table 2.2 is $O(nmT^2C)$.*

Proof Similarly to the proof of observation 2.1.1, we need to count the number of pops at line 5, which are bounded by the number of pushes at line 21. For a given pair (t,i) only one label with minimum cost is kept. On the other hand, there exist $O(nC)$ possible cost improvements for the given pair

Table 2.2: Forward Labeling Approach for SPPRC Applying Dominance Rule in Two Directions over All Eligible Labels

```
1   for each i ∈ N do
2     L(i) ← ∅; (L(i) represents all the assigned labels at node i)
3   Create an initial label at o+ and put a copy in L(o+) and L;
4   Create label L* at o− with an infinite cost;
5   while L is not empty, pop the first label L in L do
6     for each j ∈ A(i^L) do
7       if arc extension i^L to j is feasible do
8         if arc extension j to o− is feasible do
9           create label L′ at node j;
10          if i^{L′} ! = o− do
11            bool isDominated ← false;
12            for each L″ ∈ L(j) do
13              if L″ ⪯ L′ do
14                isDominated ← true;
15                break;
16              else if L′ ≺ L″ do
17                L(i^{L′})\ = L″;
18                L\ = L″;
19            if !isDominated do
20              L(i^{L′}).pushback(L′);
21              L.pushback(L′);
22          else if i^{L′} == o− do
23              if c^{L′} < c^{L*} do
24                L* ← L′;
25  return L*;
```

Table 2.3: Forward Labeling Approach for SPPRC Applying Partial Dominance Rule in One Direction over a Subset of Eligible Labels

```
1   for each  i ∈ N and t ∈ {1, ..., T}  do
2     L(t, i) ← ∅; (L(t, i) is a list of labels at node i and time t)
3   Create an initial label at (e_{o+}, o+) and put a copy in L(e_{o+}, o+) and L;
4   Create label L* with infinite cost at o−;
5   while L is not empty, pop the first label L in L  do
6     for each  j ∈ A(i^L)  do
7         if  arc extension i^L to j is feasible  do
8           if  arc extension j to o− is feasible  do
9             create label L′ at node j;
10            if  i^{L′}! = o−  do
11              bool isDominated ← false;
12              for each  L″ ∈ L(t^{L′}, i^{L′})  do
13                if  L″ ⪯ L′  do
14                  isDominated ← true;
15                  break;
16              if !isDominated  do
17                L(t^{L′}, i^{L′}) ← L′;
18                L.pushback(L′);
19            else if  i^{L′} == o−  do
20                if  c^{L′} < c^{L*}  do
21                  L* ← L′;
22  return  L*;
```

(t, i). Hence, for a given node i there exist $O(nTC)$ pushes. When a label is poped at a given node, all the nodes belonging to its adjacency list are scanned (line 6). Besides, line 12 results in $O(T)$ iterations of lines 13 to 18 for a poped label. Consequently, the worst-case computational complexity of the algorithm is $\sum_{i \in N} nT^2C|A(i)| = O(nmT^2C)$.□

Applying the idea of partial dominance test over a subset of eligible labels (criteria 2.24 to 2.26), give rise to an improved worst-case computational complexity as shown in the next observation.

Observation 2.2.2 *The worst-case computational complexity of the algorithm proposed in Table 2.2 can be improved to $O(nmTC)$ if a partial dominance test is performed over a subset of eligible labels using criteria (2.24) to (2.26) as done in the algorithm proposed in Table 2.3.*

Proof This observation can be proved in the same way as observation 2.2.1. However, in line 12 of Table 2.3, one only scans $\mathcal{L}(\mathrm{t}^{L'}, \mathrm{i}^{L'})$ which has size 1, instead of size $O(T)$ for $\mathcal{L}(\mathrm{i}^{L'})$. Hence, a saving of $O(T)$ operations resulting from applying the dominance rule over a subset of eligible labels is obtained.□

Observation 2.2.3 *Assume that $\mathcal{L}(t, i)$ is a container which gives a direct access to the generated label at node $i \in N$ and time $t \in \{1, ..., T\}$. Processing the pairs (t, i) in lexicographic order, the worst-case computational complexity of the algorithm as proposed in Table 2.3 is $O(mT)$.*

Proof The proof is similar to the one presented for observation 2.1.2. Note that in this case, $\mathcal{L}$ is a two dimensional vector of labels: one dimension corresponds to time, t, and the other dimension corresponds to network nodes, i. Hence, for each t there exists a column containing a copy of network nodes. For each node of such column, we need to check its adjacency list. Thus, for that column, we have $O(m)$ operations. Besides, a pop can happen only once for a given column as time increases in a transition. Since there exist $O(T)$ columns, the worst-case computational complexity of the algorithm will be $O(mT)$.□

It is straightforward to note that replacing time window constraints with capacity constraints leads to a similar analysis.

2.3. The Elementary Shortest Path Problem with Time Windows

The elementary shortest path problem with time windows (ESPPTW), which was first proposed by Feillet et al. [19] for graphs that contain negative cost cycles, is a shortest path problem with time windows where each node may be visited at most once. The mathematical formulation for the elementary shortest path problem with time windows is as follows:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.27}$$

s.t

$$\sum_{j\in N\setminus\{o^+,o^-\}} x_{o^+j} = 1 \tag{2.28}$$

$$\sum_{j\in N\setminus\{o^+\}} x_{ij} - \sum_{j\in N\setminus\{o^-\}} x_{ji} = 0 \quad \forall i \in N \setminus \{o^+, o^-\} \tag{2.29}$$

$$\sum_{i\in N\setminus\{o^+,o^-\}} x_{io^-} = 1 \tag{2.30}$$

$$t_i + t_{ij} - M(1 - x_{ij}) \leq t_j \quad \forall i \in N \setminus \{o^-\}, j \in N \setminus \{o^+\} \tag{2.31}$$

$$y_i = \sum_{j\in N\setminus\{o^+\}} x_{ij} \quad \forall i \in N \tag{2.32}$$

$$e_i\, y_i \leq t_i \leq \bar{l}_i\, y_i \tag{2.33}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{2.34}$$

$$y_i \in \{0,1\} \quad \forall i \in N \tag{2.35}$$

where, similarly to the SPPTW, the x_{ij} variables represent the flow in the network, t_i variables stand for the time of arrival at node i (or more generally a resource consumption up to node i), y_i is a measure to check if node i is visited, t_{ij} is the travel time and M is a big number. Constraints (2.32) limiting the number of outgoing arcs to zero or one that enforce elementarity while time windows are not tight enough.

In the definition of a label for the forward labeling approach, one needs to introduce new resources to model the concept of elementary path:

- $\mathcal{V}^L$ records the set of visited nodes for the partial elementary path

associated with label L.

Then, label L can be extended to node $j \in A(\mathrm{i}^L)$ if the following condition is satisfied on top of conditions (2.19) and (2.20):

$$j \notin \mathcal{V}^L \tag{2.36}$$

If there exists a feasible arc extension (i^L, j), then one creates a new label L' at node j as follows:

$$\begin{aligned}
\mathrm{i}^{L'} &= j && (2.37)\\
\mathrm{c}^{L'} &= \mathrm{c}^L + c_{\mathrm{i}^L j} && (2.38)\\
\mathrm{t}^{L'} &= \max\{e_j, \mathrm{t}^L + t_{\mathrm{i}^L j}\} && (2.39)\\
\mathcal{V}^{L'} &= \mathcal{V}^L \cup \{j\} && (2.40)\\
\mathrm{p}^{L'} &= L && (2.41)
\end{aligned}$$

Then, the dominance rule for L' and L'' takes the form: $L' \preceq L''$ if the all following conditions are satisfied,

$$\begin{aligned}
\mathrm{i}^{L'} &= \mathrm{i}^{L''}, && (2.42)\\
\mathrm{c}^{L'} &\leq \mathrm{c}^{L''}, && (2.43)\\
\mathrm{t}^{L'} &\leq \mathrm{t}^{L''}, && (2.44)\\
\mathcal{V}^{L'} &\subseteq \mathcal{V}^{L''}. && (2.45)
\end{aligned}$$

To verify the validity of the above dominance rule, let $\mathcal{P}$ be a partial path extending L'' to o^-. Clearly, the nodes of $\mathcal{P}$ belongs to $N \setminus \mathcal{V}^{L''}$, where $N \setminus \mathcal{V}^{L''} \subseteq N \setminus \mathcal{V}^{L'}$. Hence, $\mathcal{P}$ is a feasible extension path to extend L' to o^- with an overall cost that is smaller for $(L', \mathcal{P})$. Thus, L' dominates L''.

Such complex dominance relationship can be time consuming to check. Hence, in practice, one might consider a more restricted set of dominance conditions that model sufficient conditions for dominance that are easier to check. However, using a more restricted set of dominance conditions has the drawback of carrying more partial paths, as it does not allow the detection of all dominations between labels. In Table 2.4, we summarize different variants of the dominance rule for ESPPTW: Dom1 is the exact dominance rule that

Table 2.4: Different Variants of Dominance Rule for ESPPTW

Dom1	Dom2	Dom3
$\mathrm{i}^{L'} = \mathrm{i}^{L''}$	$\mathrm{i}^{L'} = \mathrm{i}^{L''}$	$\mathrm{i}^{L'} = \mathrm{i}^{L''}$
$\mathrm{t}^{L'} \leq \mathrm{t}^{L''}$	$\mathrm{t}^{L'} = \mathrm{t}^{L''}$	$\mathrm{t}^{L'} = \mathrm{t}^{L''}$
$\mathrm{c}^{L'} \leq \mathrm{c}^{L''}$	$\mathrm{c}^{L'} \leq \mathrm{c}^{L''}$	$\mathrm{c}^{L'} \leq \mathrm{c}^{L''}$
$\mathcal{V}^{L'} \subseteq \mathcal{V}^{L''}$	$\mathcal{V}^{L'} \subseteq \mathcal{V}^{L''}$	$\mathcal{V}^{L'} = \mathcal{V}^{L''}$

allows to eliminate more labels, while Dom2 and Dom3 are restrictions of the dominance test that do not allow to recognize all dominance relationships, but are computationally cheaper to test.

To apply the previously discussed forward labeling approaches of Tables 2.2 and 2.3, we simply need to consider that these algorithms are now using the feasibility test and dominance test described in this section. The following two observations discuss the resulting computational complexity of the algorithms of Tables 2.2 and 2.3 for this case.

Observation 2.3.1 *If graph G does not contain any negative cost cycle, then the worst-case computational complexity of the algorithm proposed in Table 2.2 is $O(2^n nmT^2C)$, when Dom1 is used.*

Proof As shown in observation 2.2.1, not generating elementary paths results in $O(nmT^2C)$ operations. Generating elementary paths adds $O(2^n)$ state differentiation for each label. Hence, the worst-case computational complexity of the whole algorithm is $O(2^n nmT^2C)$. □

Observation 2.3.2 *Using a direct access container $\mathcal{L}(t, \mathcal{V}, i)$ and processing $(t, \mathcal{V}, i)$ in lexicographic order, the worst-case computational complexity of the proposed algorithm in Table 2.3 is $O(2^n mT)$.*

Proof The proof is similar to the one presented for observation 2.2.3. However, in this case $\mathcal{L}$ is a three dimensional vector of labels: first dimension corresponds to time, second dimension corresponds to the set of visited nodes, and the third dimension corresponds to network nodes. For each $(t, \mathcal{V})$, there exists a column containing a copy of network nodes. For each node of such column we need to check its adjacency list. Thus, for that column, we have $O(m)$ operations. Besides, a pop can happen only once for a given column as time increases in a transition. Since there exist $O(2^n T)$ columns, the worst-case computational complexity of the algorithm will be $O(2^n mT)$.□

To reduce the worst-case computational complexity, one can apply the idea of partial elementary routes proposed by Baldacci et al. [3]. The idea of the so-called "ng-path" scheme is to perform a partial check for elementarity: considering that nodes that are not neighbors of the last included nodes are less likely to be candidates for inclusion in the partial solution; therefore, there is no need to maintain a data structure recording their presence in the current partial path. For label L, the subset of nodes for which we wish to check elementarity can be defined as follows:

$$\Pi^L = \{i_k \in \mathcal{V}^L \setminus \{\mathrm{i}^L\} : i_k \in \bigcap\nolimits_{s=k+1}^{p} N_{i_s}\} \cup \{\mathrm{i}^L\} \tag{2.46}$$

where p is the position of node i^L along the partial path and i_k represents node i anterior in the sequence by k position, and $N_i \subseteq N$ defines the neighborhood of $i \in N$, also called "ng-set". The neighborhood may be defined for instance as the $|N_i|$ closest nodes to i, including i itself [45]. The walk created in this way is called 'ng-route".

The use of such an idea in forward labeling approach requires to replace the set of visited nodes $\mathcal{V}^L$ with Π^L in the definition of a label. Then, a feasible arc extension is defined by conditions (2.19) and (2.20), but condition (2.36) is replaced by

$$j \notin \Pi^L \tag{2.47}$$

If such a feasible arc extension does exist, then a new label can be created at node j as follows:

$$\begin{aligned}
\mathrm{i}^{L'} &= j && (2.48)\\
\mathrm{c}^{L'} &= \mathrm{c}^L + c_{\mathrm{i}^L j} && (2.49)\\
\mathrm{t}^{L'} &= \max\{e_j, \mathrm{t}^L + t_{\mathrm{i}^L j}\} && (2.50)\\
\Pi^{L'} &= \Pi^L \cap N_j \cup \{j\} && (2.51)\\
\mathrm{p}^{L'} &= L && (2.52)
\end{aligned}$$

In this case, different variants of dominance rule can be expressed in the same way as the one for ESPPTW, while the set of visited nodes $\mathcal{V}^L$ is replaced by Π^L.

Observation 2.3.3 *Using a direct access container $\mathcal{L}(t, \Pi, i)$ and processing (t, Π, i) in lexicographic order, the worst-case computational complexity of the proposed algorithm in Table 2.3 is $O(2^{|N_i|}mT)$.*

Proof As the worst-case computational complexity of the shortest path problem with time windows (under similar assumption) is $O(mT)$ and the ng-set, N_i, needs $O(2^{|N_i|})$ additional operations, the worst-case computational complexity of the proposed algorithm in Table 2.3 is $O(2^{|N_i|}mT)$. □

2.4. The Elementary Shortest Path Problem with Time Windows and Pickup and Delivery

The elementary shortest path problem with time windows and pickup and delivery (ESPPTWPD) is another variants of shortest path problems with resource constraint (SPPRC). In this problem, the goal is to serve a set of requests while respecting the predefined time windows. Let $\mathcal{R}$ denote the set of requests. Each request $r \in \mathcal{R}$ contains a pickup node p_r and a delivery node d_r. For each such node a time window is defined. To cover a request, first its pickup node must be served then its delivery node. This constraint is called precedence constraint. The mathematical formulation of ESPPTWPD is defined over graph $G = (N, A)$ where $N = P \cup D \cup \{o^+, o^-\}$, P is the set

of pickup nodes and D is the set of delivery nodes.

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.53}$$

s.t

$$\sum_{j\in P} x_{o^+j} = 1 \tag{2.54}$$

$$\sum_{j\in N\setminus\{o^+\}} x_{ij} - \sum_{j\in N\setminus\{o^-\}} x_{ji} = 0 \quad \forall i \in N \setminus \{o^+, o^-\} \tag{2.55}$$

$$\sum_{i\in D} x_{io^-} = 1 \tag{2.56}$$

$$t_i + t_{ij} - M(1 - x_{ij}) \leq t_j \quad \forall i \in N \setminus \{o^-\}, j \in N \setminus \{o^+\} \tag{2.57}$$

$$y_r = \sum_{j\in N\setminus\{o^+,o-\}} x_{p_r j} \quad \forall r \in \mathcal{R} \tag{2.58}$$

$$y_r = \sum_{i\in N\setminus\{o^+,o-\}} x_{id_r} \quad \forall r \in \mathcal{R} \tag{2.59}$$

$$e_{p_r}\, y_r \leq t_{p_r} \leq \bar{l}_{p_r}\, y_r \quad \forall r \in \mathcal{R} \tag{2.60}$$

$$e_{d_r}\, y_r \leq t_{d_r} \leq \bar{l}_{d_r}\, y_r \quad \forall r \in \mathcal{R} \tag{2.61}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{2.62}$$

$$y_r \in \{0, 1\} \quad \forall r \in \mathcal{R} \tag{2.63}$$

where similarly, the x_{ij} variables represent the flow in the network, t_i variables stand for the time of arrival at node i, y_r is a measure to check if request r is serviced, t_{ij} is the travel time, and M is a big number.

This problem was first introduced by Sol [41] in the context of the pickup and delivery problem with time windows. Later Sigurd et al. [40] used it with additional precedence constraints. In 2009, Robke and Cordeau [38] proposed a new labeling algorithm that encompass a stronger dominance rule with respect to the algorithm proposed by Sol [41] and the general one described by Sigurd et al. [40].

To solve the problem using the forward labeling approach a new resource must be added to the features of the defined label for ESPPTW:

- $\mathcal{O}^L$, which is the set of open requests. This set contains all the requests whose pickup node is served, but not the delivery node.

A feasible arc extension from label L to node j includes condition (2.19)

together with some additional conditions outlined below. To discuss further feasibility conditions, two different cases are considered:

Case 1: If j is a pickup node, $p_r \in P$, then the additional feasibility conditions are:

$$r \notin \mathcal{V}^L \tag{2.64}$$
$$r \notin \mathcal{O}^L \tag{2.65}$$

where $r \notin \mathcal{V}^L$ means neither p_r nor d_r are served.

If the above conditions are satisfied, then the possibility to reach node o^- must be verified to find out early the probable infeasibility of the label at node j in further extensions. Let H be the set of all paths starting at node j and ending up at node o^-, while the delivery node of all open requests at node j are the intermediate nodes of these paths. In other words, different permutations of these delivery nodes generate the elements of H, where condition (2.19) is the only measure for the feasibility of these paths. There exist different possibilities to generate the elements of H. One way is enumeration. Considering the combinatorial structure of H, it is better either not to provide all the elements of H by enumeration or to apply an approximate approach. Checking that there exists a feasible completion using one of the element of H is a necessary condition for feasibility, assuming triangular inequality of the transition times.

If all the mentioned necessary feasibility conditions are satisfied, then label, L', can be created at node j as follows:

$$\mathrm{i}^{L'} = j \tag{2.66}$$
$$\mathrm{c}^{L'} = \mathrm{c}^L + c_{\mathrm{i}^L j} \tag{2.67}$$
$$\mathrm{t}^{L'} = \max\{e_j, \mathrm{t}^L + t_{\mathrm{i}^L j}\} \tag{2.68}$$
$$\mathcal{O}^{L'} = \mathcal{O}^L \cup \{r\} \tag{2.69}$$
$$\mathrm{p}^{L'} = L \tag{2.70}$$

Case 2: If j is a delivery node, $d_r \in D$, then the additional feasibility

condition to condition (2.19) is:

$$r \in \mathcal{O}^L \tag{2.71}$$

If conditions (2.19) and (2.71) are satisfied, then the possibility to reach node o^- must be verified similar to the one explained for the first case. Having satisfied all feasibility conditions, label L' is:

$$\begin{aligned}
\mathrm{i}^{L'} &= j && (2.72)\\
\mathrm{c}^{L'} &= \mathrm{c}^L + c_{\mathrm{i}^L j} && (2.73)\\
\mathrm{t}^{L'} &= \max\{e_j, \mathrm{t}^L + t_{\mathrm{i}^L j}\} && (2.74)\\
\mathcal{V}^{L'} &= \mathcal{V}^L \cup \{r\} && (2.75)\\
\mathcal{O}^{L'} &= \mathcal{O}^L \setminus \{r\} && (2.76)\\
\mathrm{p}^{L'} &= L && (2.77)
\end{aligned}$$

The possible variations of dominance rule for this case are then as follows:

Dom'1:

$$\begin{aligned}
\mathrm{i}^{L'} &= \mathrm{i}^{L''} && (2.78)\\
\mathrm{t}^{L'} &\leq \mathrm{t}^{L''} && (2.79)\\
\mathrm{c}^{L'} &\leq \mathrm{c}^{L''} && (2.80)\\
\mathcal{V}^{L'} &\subseteq \mathcal{V}^{L''} && (2.81)\\
\mathcal{O}^{L'} &\subseteq \mathcal{O}^{L''} && (2.82)
\end{aligned}$$

and

Dom'2:

$$\begin{aligned}
\mathrm{i}^{L'} &= \mathrm{i}^{L''} && (2.83)\\
\mathrm{t}^{L'} &= \mathrm{t}^{L''} && (2.84)\\
\mathrm{c}^{L'} &\leq \mathrm{c}^{L''} && (2.85)\\
\mathcal{V}^{L'} &\subseteq \mathcal{V}^{L''} && (2.86)\\
\mathcal{O}^{L'} &\subseteq \mathcal{O}^{L''} && (2.87)
\end{aligned}$$

Dom'1 was proposed by Robke and Cordeau [38]. In their work, they

mentioned that such dominance rule is valid only if the coefficients of the cost matrix follows the triangle inequality. Indeed, a feasible extension $\mathcal{P}''$ for L'' can give rise to a feasible extension $\mathcal{P}'$ to extend L' to o^- by deleting the delivery nodes of the open requests belonging to $\mathcal{O}^{L''} \setminus \mathcal{O}^{L'}$. The overall cost of $(L', \mathcal{P}')$ is smaller than the overall cost of $(L'', \mathcal{P}'')$ if the triangle inequality is satisfied. In the case that the cost matrix does not follow the triangle inequality the subset relation in conditions (2.82) and (2.87) must be replaced by equality. For more details refer to [38].

In the following observations, the worst-case computational complexity of the associated variants of the forward labeling algorithms are discussed.

Observation 2.4.1 *If graph G does not contain any negative cost cycle, then the worst-case computational complexity of the algorithm proposed in Table 2.2 is $O(2^{|\mathcal{R}|}nmT^2C)$ when Dom'1 is applied.*

Proof As proved in observation 2.2.1, SPPTW needs $O(nmT^2C)$ operations. On the other hand, there exist at most $O(2^{|\mathcal{R}|})$ labels obtained by enumerating different possibilities for the set of open and visited requests at each node with best cost. Hence, the worst-case computational complexity of the algorithm is $O(2^{|\mathcal{R}|}nmT^2C)$. □

The worst-case computational complexity of the proposed algorithm can improve as shown in the next observations. For simplicity first we consider the non-elementary case.

Observation 2.4.2 *The worst-case computational complexity of the proposed algorithm for SPPTWPD is $O(2^{|\mathcal{R}|}mT)$ if we use $\mathcal{L}(t, \mathcal{O}, i)$ and we process $(t, \mathcal{O}, i)$ in lexicographical order.*

Proof The proof is similar to the one presented for observation 2.3.2, while the set of visited requests in that case must be replaced by the set of open requests.□

Observation 2.4.3 *The worst-case computational complexity of the proposed algorithm for ESPPTWPD is $O(2^{|\mathcal{R}|}mT)$ if $\mathcal{L}(t, \mathcal{V}, \mathcal{O}, i)$ is applied and $(t, \mathcal{V}, \mathcal{O}, i)$ are processed in lexicographical order.*

Proof Applying elementary routes add $O(2^{|\mathcal{R}|})$ labels to $O(2^{|\mathcal{R}|}mT)$ operations of non-elementary case proved in former observation. Thus, the worst-case computational complexity of the whole algorithm is $O(2^{|\mathcal{R}|}mT)$.□

2.5. The Elementary Shortest Path Problem with Time Windows, Pickup and Delivery, Load, and Ride Time Constraints

We now consider the one-vehicle problem that actually arises as a subproblem in solving the dial-a-ride problem (as formulated in Section 1.4.1) via a Lagrangian relaxation approach. The discussion herein is valid under Assumption 5-c. If constraint (1.5) can be violated at price π_i for each $i \in P'$, the resulting price collecting subproblem, for a given vehicle $k \in K$, takes the form:

$$\min \quad \sum_{i \in N} \sum_{j \in N} (c_{ij} - \pi_i) x_{ij} \tag{2.88}$$

s.t

$$y_r = \sum_{j \in N \setminus \{o^+\}} x_{p_r j} \quad \forall r \in \mathcal{R} \tag{2.89}$$

$$y_r = \sum_{i \in N \setminus \{o^-\}} x_{i d_r} \quad \forall r \in \mathcal{R} \tag{2.90}$$

$$\sum_{j \in P'} x_{o^+ j} = 1 \tag{2.91}$$

$$\sum_{j \in N \setminus \{o^-\}} x_{ji} - \sum_{j \in N \setminus \{o^+\}} x_{ij} = 0 \quad \forall i \in N \setminus O' \tag{2.92}$$

$$\sum_{i \in D'} x_{i o^-} = 1 \tag{2.93}$$

$$t_i + s_i + t_{ij} - M_{ij}(1 - x_{ij}) \leq t_j \quad \forall i \in N \setminus \{o^-\}, j \in N \setminus \{o^+\} \tag{2.94}$$

$$Q_i + q_j - W_{ij}(1 - x_{ij}) \leq Q_j \quad \forall i \in N \setminus \{o^-\}, j \in N \setminus \{o^+\} \tag{2.95}$$

$$e_{p_r}\, y_r \leq t_{p_r} \leq \bar{l}_{p_r}\, y_r \quad \forall r \in \mathcal{R} \tag{2.96}$$

$$e_{d_r}\, y_r \leq t_{d_r} \leq \bar{l}_{d_r}\, y_r \quad \forall r \in \mathcal{R} \tag{2.97}$$

$$w_i \geq t_i - l_i \quad \forall i \in N \setminus O' \tag{2.98}$$

$$0 \leq w_i \leq \bar{l}_i - l_i \quad \forall i \in N \setminus O' \tag{2.99}$$

$$q_r\, y_r \leq Q_{p_r} \leq Q_k\, y_r \quad \forall r \in \mathcal{R} \tag{2.100}$$

$$0 \leq Q_{d_r} \leq (Q_k - q_{d_r})\, y_r \quad \forall r \in \mathcal{R} \tag{2.101}$$

$$t_{d_r} - t_{p_r} - s_{p_r} \leq f_r \quad \forall r \in \mathcal{R} \tag{2.102}$$

$$t_{p_r d_r}\, y_r \leq f_r \leq b_r^f\, y_r \quad \forall r \in \mathcal{R} \tag{2.103}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{2.104}$$

$$y_r \in \{0, 1\} \quad \forall r \in \mathcal{R} \tag{2.105}$$

The above formulation is an ESPPTWPD with some additional con-

straints: load constraint, ride time constraint, and waiting time constraint. We name this new pricing problem as an Elementary Shortest Path Problem with Time Windows, Pickup and Delivery, Load, and Ride Time Constraint (ESPPTWPDLR). This pricing problem involves a core subproblem that allows to characterize a feasibility set for timing decisions once a sequence is fixed. Before outlining our forward labeling approach that solves the ESPPTWPDLR, we present the timing subproblem.

2.5.1 The Timing Subproblem

The material of sections 2.5.1-2.5.3 is extracted from Detienne et al. [17]. A feasibility set of timing decisions for a given sequence, S, can be characterized through the following linear program:

$$\begin{aligned}
t_j &\geq t_i + t_{ij} + s_i \quad \forall (i,j) : x_{ij} = 1 && (2.106)\\
e_i \leq t_i &\leq \bar{l}_i \quad \forall i \in N' && (2.107)\\
t_{d_r} - t_{p_r} - s_{p_r} &\leq f_r \quad \forall r \in \mathcal{R}' && (2.108)\\
w_i &\geq t_i - l_i \quad \forall i \in N' && (2.109)\\
0 \leq w_i &\leq \bar{l}_i - l_i \quad \forall i \in N' && (2.110)\\
t_{p_r d_r} \leq f_r &\leq b_r^f \quad \forall r \in \mathcal{R}' && (2.111)
\end{aligned}$$

Note that S is associated with an elementary path that is feasible in terms of conditions (2.89)-(2.101); however, it might be infeasible regarding ride time constraints. Besides, N' and $\mathcal{R}'$ denote the set of visited nodes and the set of visited requests along the sequence, respectively.

Conditions (2.109) and (2.110) do not really participate in defining the feasibility set. Similarly, the value of f_r variables can only affect the value of objective function if we assume that $c_r^f > 0$. Hence, the following three constraints are enough to define the feasibility set:

$$\begin{aligned}
t_j &\geq t_i + t_{ij} + s_i \quad \forall (i,j) : x_{ij} = 1 && (2.112)\\
e_i \leq t_i &\leq \bar{l}_i \quad \forall i \in N' && (2.113)\\
t_{d_r} - t_{p_r} - s_{p_r} &\leq b_r^f \quad \forall r \in \mathcal{R}' && (2.114)
\end{aligned}$$

The above three constraints define the feasibility set for timing decisions

via inducing tighten lower and upper bounds on the t_i values, where $i \in N'$. The general idea of tightening time windows is to consider the lower bound and the upper bound of time windows separately. The procedure how to tighten time windows is explained more precisely in the next section, where we assume that the nodes are numbered according to their position in the sequence, $i = 1, \ldots, |N'|$.

2.5.2 Checking Feasibility via Tightening Time Windows

Adjusting earliest starting times. Constraints (2.112) and (2.114) together with the lower bound on the value of start service times (first inequality of constraint (2.113)) induce tighter lower bounds on t_i values. Such tighter earliest start service times are the solution of the following optimization problem:

$$\min \quad \sum_{i \in N'} (t_i - t_{\text{begin}}) \tag{2.115}$$

$$t_j \geq t_i + t_{ij} + s_i \quad \forall (i,j) : x_{ij} = 1 \tag{2.116}$$

$$t_{p_r} \geq t_{d_r} - s_{p_r} - b_r^f \quad \forall r \in \mathcal{R}' \tag{2.117}$$

$$t_i \geq t_{\text{begin}} + e_i \quad \forall i \in N' \tag{2.118}$$

where t_{begin} can be set to 0: it represents the start time of the sequence.

This problem is a scheduling problem with generalized precedence constraint that can be solved by a dual approach, computing longest path in the associated graph G' (as illustrated in Figure 2.1, and the numerical example in 2.2) from node "*begin*" representing the beginning of the sequence. If the associated graph has a positive cost cycle the problem is infeasible. Otherwise, the longest path lengths, which are noted as η_i, define updated earliest start times: $e'_i = \eta_i$. They are computed standardly using recursion:

$$\eta_j = \max\{\eta_j, \max_{(i,j) \in G'} \{\eta_i + l_{ij}\}\} \text{ for } j = 1, \ldots, |N'| \tag{2.119}$$

where $(i,j) \in G'$ are the arcs of the graph (such as illustrated in Figure 2.1), l_{ij} is the time lag on the arc, and the nodes are considered (numbered) in the order in which they appear in the sequence. The recursion is initialized with $\eta_i = e_i$. The recursion can be applied for each node as long as an

update occurs. If a distance is updated on iteration $|N'|$, the graph contains a positive cost cycle.

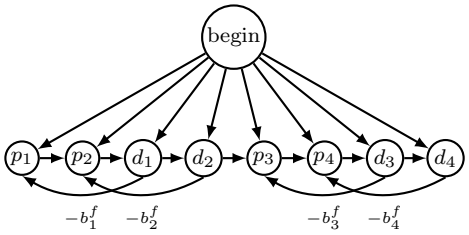


Figure 2.1: Earliest starting times are determined by the longest path from "begin".

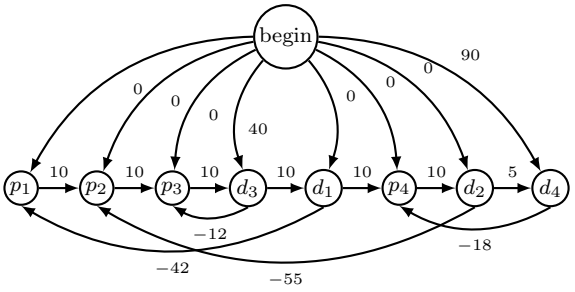


Figure 2.2: Example 1.

Adjusting latest starting times. Symmetrically, constraints (2.112) and (2.114) together with the upper bound on the value of start service times (second inequality of constraint (2.113)) induce tighter upper bounds on t_i values, where $i \in N'$. Assuming a bound T on the time horizon, the latest start service times can be obtained from the solution of the following minimization problem modeling earliest start service times from T (reverse time):

$$\min \quad \sum_{i \in N'} (t_i - t_{\text{end}}) \tag{2.120}$$

$$t_i \geq t_j + t_{ij} + s_i \quad \forall (i,j) : x_{ij} = 1 \tag{2.121}$$

$$t_{d_r} \geq t_{p_r} - s_{p_r} - b_r^f \quad \forall r \in \mathcal{R}' \tag{2.122}$$

$$t_i \geq t_{\text{end}} + (T - \bar{l}_i) \quad \forall i \in N' \tag{2.123}$$

where t_{end} can be set to 0; it stands for the reverse starting time that is counted from the end of the sequence. Similarly, the minimum reverse starting time problem is a scheduling problem with generalized precedence constraint that can be solved by a dual approach, computing longest paths in the associated graph (such as illustrated in Figure 2.3) from node "*end* " representing the end of the route. If the associated graph has a positive cost cycle, the problem is infeasible. Otherwise, the longest path lengths, η_i, are computed using recursion (2.119) but numbering the nodes in reverse order of the sequence S. The length of the longest path leads to define updated latest start times: $\bar{l}'_i = T - \eta_i$.

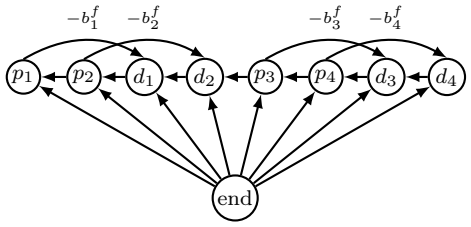


Figure 2.3: Latest starting times are determined by the longest path from "end".

Having preprocessed lower and upper bounds, necessary feasibility conditions are:

$$e'_i \leq \bar{l}'_i \quad \forall i \in N'$$

The above condition is not only a necessary feasibility condition but also a sufficient one although updated earliest and latest start service times have been computed separately. Indeed, for each $i \in N'$, constraints $t_{\text{begin}} \geq t_i - \bar{l}_i$ could have been added to model (2.115)-(2.118). However, such constraint can only be active if $e'_i \geq \bar{l}_i$, i.e. if the timing problem is infeasible; however, the infeasibility of the timing problem can be naturally detected by the above feasibility test. Symmetrically, for each $i \in N'$, constraints $t_{\text{end}} \geq t_i - (T - e_i)$ could have been added to model (2.121)-(2.123), but they are redundant if $\bar{l}'_i \geq e_i$, which is a condition of the feasibility check that we apply to the overall timing problem.

Considering Assumption 5-c allows us to set timing decisions to updated earliest start service times if the feasibility set for timing decisions is not

empty. Hence, the minimum reverse service start time values are only needed to guarantee the feasibility of the timing decisions and the value of updated latest start service times do not affect the optimal value of earliest start service times. Such assumption also leads us to decompose the timing problem into **independent subsequences** that do not share any request.

Definition 1 Sequence decomposition
A sequence S of nodes defining a route can be partitioned into subsequences $S_1, \ldots, S_K$ such that the vehicle is empty on leaving the last node (denoted as j_k) of subsequence $S_k, k = 1, \ldots, K$.

In Figure 2.1, there exist two subsequences: p_1 to d_2 on one hand, and p_3 to d_4 on the other hand. Note that there is no path from "begin" to d_2 going through a node of the second subsequence (S_2). Hence, the timing decisions of the nodes beyond d_2 have no impact on the timing decisions of node d_2 and prior nodes. This characteristic allows us to first name the first subsequence as an independent subsequence. Then the impact of the timing decisions of the nodes of this independent subsequence on the earliest starting times of the nodes of the second subsequence can be summarized through an arc , as illustrated in Figure 2.4: its lag is $e'_{d_2} + s_{d_2} + t_{d_2\, p_3}$. Equivalently, one can redefine $e_{p_3} = \max\{e_{p_3}, e'_{d_2} + s_{d_2} + t_{d_2\, p_3}\}$.

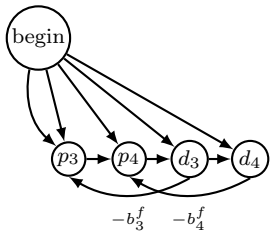


Figure 2.4: Earliest starting times aggregating the first subsequence into an arc.

Definition 2 Open subequence
We name the last independent subsequence of S that is not closed yet as an open subsequence.

Suppose that S_{k+1} is the open subsequence of sequence S. The feasibility of the timing problem after adding node i to sequence S can be checked

by solving the minimum starting time problem and the minimum reverse starting time problem on S_{k+1} separately. As it was mentioned previously, the updated latest start service times do not impact on the optimal solution (t^*), of the timing problem (2.115)-(2.118) for any node in subsequences $S_1, \ldots, S_k$ unless arc extension (j_k, i_{k+1}) becomes infeasible because $e_{j_k} + t_{j_k\, i_{k+1}} + s_{j_k} > \bar{l}'_{i_{k+1}}$, where j_k is the last node of subsequence S_k and i_{k+1} is the first node of subsequence S_{k+1} (i.e. adding node i to S_{k+1} makes the timing problem infeasible). Hence, latest start service times of the nodes of S_{k+1} are not updated for their own sake, but only for the sake of checking if an update can impact the feasibility of the problem. By initializing $\bar{l}' = t^*$, we can observe whether these values can be updated using the longest path procedure from the "end" node or not. However, the propagation will stop at node j_k, unless $e_{j_k} + s_{j_k} + t_{j_k\, i_{k+1}} > \bar{l}'_{i_{k+1}}$. For more details refer to [17].

2.5.3 Residual Problem and Dominance Rule

To solve the ESPPTWPDLR using similar forward labeling approach presented for the ESPPTWPD, we redefine the labels on a different state space. Each label L contains two fields: defining attributes and auxiliary attributes. Defining attributes encompass the quantity consumption of resources in the partial path associated to label L, while auxiliary attributes are quantities that could be computed using the defining attributes. The aim of introducing auxiliary attributes is to ease computations by avoiding to recompute quantities from scratch.

Defining Attributes:

- $\mathrm{i}^L \in N$ is the last node of the partial path defining the label;
- e^L is the vector of updated earliest start service times;
- $\bar{l}^L$ is the vector of updated latest start service times;
- c^L is the accumulated cost;
- $\mathcal{V}^L$ is the set of visited requests;
- $\mathcal{O}^L$ is the set of open requests;
- S^L is the open subsequence of nodes of the partial path;

- p^L is a pointer to the predecessor label of L;

Auxiliary Attributes:

- $\mathrm{j}^L \in N$ is the end node of the last independent subsequence (that is closed) of the partial path prior to S^L;
- q^L is the load of the vehicle at the end of the partial path;
- r^L is the index of the request corresponding to the node i^L;

Now we outline the feasibility conditions to extend an arbitrary label L to a node $j \in N$ ($j \neq \mathrm{i}^L$). To do so, three different cases are considered.

Case 1 : Assume that j is a pickup node, $j = p_r \in P'$. The feasibility conditions are then as follows:

i : $\mathrm{e}^L_{\mathrm{i}^L} + s_{\mathrm{i}^L} + t_{\mathrm{i}^L j} \leq \bar{l}_j$.

ii : $\mathrm{q}^L + q_j \leq Q_k$.

iii : $r \notin \mathcal{V}^L$.

iv : $r \notin \mathcal{O}^L$.

Constraints (i) and (ii) covers the feasibility conditions for time window and load constraints, respectively. Conditions (iii) and (iv) express the fact that arc extension (i^L, j) can happen only if request r has not been visited nor opened yet. Note that there is no need to consider any feasibility condition for customer waiting time constraints, as considering Assumption 5-c allows us to set start service times to the earliest possible values.

In addition to the above necessary feasibility conditions, there exist some other feasibility criteria that allow an early detection of infeasibility of the potential label at node j. These conditions are discussed after outlining the necessary conditions for cases 2 and 3.

Case 2 : Now let j be a delivery node, $j = d_r \in D'$. The feasibility conditions of this case include the following conditions:

i : adding j leaves the timing subproblem on $< S^L, j >$ feasible.

ii : $r \in \mathcal{O}^L$.

Case 3 : In the case that $j = o^-$, the feasibility conditions are as follows:

i : $\mathrm{e}^L_{\mathrm{i}^L} + s_{\mathrm{i}^L} + t_{\mathrm{i}^L j} \leq \bar{l}_j$.

ii : $\mathcal{O} = \emptyset$.

For Case 1 and Case 2, it is possible to detect the infeasibility of the potential label at node j early. To do so, one needs to verify the existence of a feasible path from j to o^-. Let H be the set of all paths starting at node j and ending up at node o^-. To generate the elements of H an enumeration scheme is applied for the algorithms outlined in Tables 2.2 and 2.3, while time feasibility is the only measure of the feasibility. To avoid adding too many operations to the computational complexity of the algorithms, an enumeration scheme is used only when $|\mathcal{O}^L| \leq 2$.

In the case that the feasibility conditions are satisfied, a new label L' can be generated at node j as follows:

$$\mathrm{i}^{L'} = j \tag{2.124}$$
$$\mathrm{c}^{L'} = \mathrm{c}^L + c_{\mathrm{i}^L j} \tag{2.125}$$
$$\mathrm{r}^{L'} = r \tag{2.126}$$
$$\mathrm{q}^{L'} = \begin{cases} \mathrm{q}^L + q_j & \text{if } j \in P' \\ \mathrm{q}^L - q_j & \text{if } j \in D' \end{cases} \tag{2.127}$$
$$\mathcal{V}^{L'} = \begin{cases} \mathcal{V}^L & \text{if } j \in P' \\ \mathcal{V}^L \cup \{r\} & \text{if } j \in D' \end{cases} \tag{2.128}$$
$$\mathcal{O}^{L'} = \begin{cases} \mathcal{O}^L \cup \{r\} & \text{if } j \in P' \\ \mathcal{O}^L \setminus \{r\} & \text{if } j \in D' \end{cases} \tag{2.129}$$
$$\mathrm{j}^{L'} = \begin{cases} j & \text{if } |\mathcal{O}^{L'}| = 0 \\ \mathrm{j}^L & \text{if } |\mathcal{O}^{L'}| \neq 0 \end{cases} \tag{2.130}$$
$$\mathrm{S}^{L'} = \begin{cases} \emptyset & \text{if } |\mathcal{O}^{L'}| = 0 \\ \mathrm{S}^L \cup \{j\} & \text{if } |\mathcal{O}^{L'}| \neq 0 \end{cases} \tag{2.131}$$
$$\mathrm{e}^{L'}_i = \mathrm{e}^L_i \quad \forall i \in \mathrm{S}^{L'} \tag{2.132}$$
$$\bar{l}^{L'}_i = \bar{l}^L_i \quad \forall i \in \mathrm{S}^{L'} \tag{2.133}$$
$$\mathrm{p}^{L'} = L \tag{2.134}$$

Once a new label is generated a dominance rule can be applied. A partial solution L' dominates another partial solution L'', which we denote as

$$L' \prec L''$$

if any feasible completion of L'', say $\mathcal{P}_2^{L''}$, allows us to build a feasible completion of L', say $\mathcal{P}_2^{L'}$ with a better solution value, i.e.

$$c(\mathcal{P}_1^{L'}, \mathcal{P}_2^{L'}) \leq c(\mathcal{P}_1^{L''}, \mathcal{P}_2^{L''})$$

where $\mathcal{P}_1^{L'}$ and $\mathcal{P}_1^{L''}$ indicate the partial paths associated to labels L' and L'' and $c(\mathcal{P}_1^{L'}, \mathcal{P}_2^{L'})$ and $c(\mathcal{P}_1^{L''}, \mathcal{P}_2^{L''})$ refer to the cost of the paths generated by appending $\mathcal{P}_2^{L'}$ and $\mathcal{P}_2^{L''}$ to $\mathcal{P}_1^{L'}$ and $\mathcal{P}_1^{L''}$, respectively.

As it was mentioned previously, adding further nodes to partial paths $\mathcal{P}_1^{L'}$ and $\mathcal{P}_1^{L''}$ can make the relevant timing problem infeasible. However, if it is feasible, the timing decisions of the nodes that appear prior to nodes $j^{L'}$ and $j^{L''}$ do not change. Hence, if there exists any feasible path extension for $\mathcal{P}_1^{L'}$ and $\mathcal{P}_1^{L''}$, the values of $\mathrm{c}^{L'}$ and $\mathrm{c}^{L''}$ do not change.

Recall that we have made Assumption 5-c (no timing cost), which simplifies the dominance rules. Under such assumption, label L' dominates L'' ($L' \prec L''$), if the following conditions are satisfied:

$$\mathrm{i}^{L'} = \mathrm{i}^{L''} \tag{2.135}$$

$$\mathrm{e}^{L'}_{\mathrm{i}^{L'}} \leq \mathrm{e}^{L''}_{\mathrm{i}^{L''}} \tag{2.136}$$

$$\mathrm{e}^{L'}_{\mathrm{j}^{L'}} \leq \mathrm{e}^{L''}_{\mathrm{j}^{L''}} \quad \text{if } |\mathcal{O}^{L'}| = 1 \tag{2.137}$$

$$\mathrm{j}^{L'} = \mathrm{j}^{L''} \quad \text{if } |\mathcal{O}^{L'}| = 1 \tag{2.138}$$

$$\mathrm{c}^{L'} \leq \mathrm{c}^{L''} \tag{2.139}$$

$$\mathcal{V}^{L'} \subseteq \mathcal{V}^{L''} \tag{2.140}$$

$$|\mathcal{O}^{L'}| = |\mathcal{O}^{L''}| = \begin{cases} 1 & \text{if } \mathrm{i}^{L'} \in P' \\ 0 & \text{if } \mathrm{i}^{L'} \in D' \end{cases} \tag{2.141}$$

Assume that $\mathcal{P}_2^{L''}$ is a feasible completion of L''. We claim that $\mathcal{P}_2^{L''}$ is a feasible completion of L' when no timing cost is assumed. As $\mathcal{V}^{L'} \subseteq \mathcal{V}^{L''}$ and $\mathcal{P}_2^{L''}$ is an elementary path extension for label L'', it is elementary for L',

too. Suppose that $\mathrm{i}^{L'}$ (and consequently $\mathrm{i}^{L''}$) is a pickup node and $|\mathcal{O}^{L'}| = |\mathcal{O}^{L''}| = 1$. As $\mathcal{P}_2^{L''}$ covers the set of open requests at L'', it covers the set of open request at L', too. Besides, the backward propagation of latest service time cannot make $\mathcal{P}_1^{L'}$ infeasible, since $\mathrm{j}^{L'} = \mathrm{j}^{L''}$ and $\mathrm{e}^{L'}_{\mathrm{j}^{L'}} \leq \mathrm{e}^{L''}_{\mathrm{j}^{L''}}$. Hence, the timing problem remains feasible when $\mathcal{P}_2^{L''}$ is appended to $\mathcal{P}_1^{L'}$. Now suppose that $\mathrm{i}^{L'}$ (and consequently $\mathrm{i}^{L''}$) is a delivery node and $|\mathcal{O}^{L'}| = |\mathcal{O}^{L''}| = 0$. In this case, the backward propagation of latest start service times cannot make $\mathcal{P}_1^{L'}$ infeasible, since $\mathrm{e}^{L'}_{\mathrm{i}^{L'}} \leq \mathrm{e}^{L''}_{\mathrm{i}^{L''}}$ and $\mathrm{e}^{L''}_{\mathrm{i}^{L''}} + s_{\mathrm{i}^{L''}} + t_{\mathrm{i}^{L''}\,j} \leq \overline{l}'_j$, where j is the first node of $\mathcal{P}_2^{L''}$. Finally, as we do not consider any timing cost, we have

$$c(\mathcal{P}_1^{L'}, \mathcal{P}_2^{L''}) = \mathrm{c}^{L'} + c(\mathcal{P}_2^{L''}) \leq \mathrm{c}^{L''} + c(\mathcal{P}_2^{L''}) = c(\mathcal{P}_1^{L''}, \mathcal{P}_2^{L''})$$

where $c(\mathcal{P}_2^{L''})$ includes the cost of the transition from $\mathrm{i}^{L'}$ to the first node of $\mathcal{P}_2^{L''}$.

2.5.4 Heuristic Approach to Set Timing Decisions

To avoid large computational times we propose a heuristic approach to mange the ride time issue in the process of forward labeling approach. This heuristic is inspired by the idea of forward time slack proposed by Cordeau [12]. In this case, the labels of the forward labeling approach are defined as follows:

Defining Attributes:

- $\mathrm{i}^L \in N$ is the node of the label.
- $\mathcal{A}^L$ is the vector of earliest start service times. Element j of this vector noted as a^L_j, represents the earliest start service time at node j.
- c^L is the accumulated cost.
- $\mathcal{V}^L$ is the set of visited requests at the node. The service of the elements of $\mathcal{V}^L$ is completed and the set is memorized to avoid cycles.
- $\mathcal{O}^L$ is the set of open requests at the node. The service of the elements of $\mathcal{O}^L$ is not completed.

- p^L is a pointer to the predecessor label of L.

Auxiliary Attributes:

- r^L is the index of the request corresponding to the node.
- q^L is the load of the vehicle after visiting the node.

The feasibility conditions for an arc extension from label L to a node $j \in N$ ($j \neq \mathrm{i}^L$) are then as follows:

Case 1 : Assume that j is a pickup node, $j = p_r \in P'$. The feasibility conditions are then as follows:

$$\begin{aligned} \mathrm{a}^L_{\mathrm{i}^L} + s_{\mathrm{i}^L} + t_{\mathrm{i}^L j} &\leq \bar{l}_j && (2.142)\\ \mathrm{q}^L + q_j &\leq Q_k && (2.143)\\ r &\notin \mathcal{V}^L && (2.144)\\ r &\notin \mathcal{O}^L && (2.145) \end{aligned}$$

Similarly, Constraints (2.142) and (2.143) covers the feasibility conditions for time window and load constraints, respectively. Conditions (2.144) and (2.145) express the fact that arc extension (i^L, j) can happen only if request r has not been visited nor opened yet.

Case 2 : Now let j be a delivery node, $j = d_r \in D'$. The feasibility conditions of this case include condition (2.142) together with the following conditions:

$$\begin{aligned} r &\in \mathcal{O}^L && (2.146)\\ \max\{e_j, \mathrm{a}^L_{\mathrm{i}^L} + s_{\mathrm{i}^L} + t_{\mathrm{i}^L j}\} - \mathrm{a}^L_{p_r} - s_{p_r} &\leq b^f_r && (2.147) \end{aligned}$$

where b^f_r is the maximum value for the ride time. Conditions (2.146) and (2.147) cover the precedence constraint and the ride time constraint, respectively.

Case 3 : In the case that $j = o^-$, condition (2.142) together with the following condition must be valid:

$$\mathcal{O}^L = \emptyset \qquad (2.148)$$

It is possible to apply further feasibility conditions to detect the infeasibility of the potential label at node j early, as it was explained previously.

In the case that the feasibility conditions are satisfied, the new label L' can be generated at node j as follows:

$$\mathrm{i}^{L'} = j; \tag{2.149}$$

$$\mathrm{a}_j^{L'} = \max\{e_j, \mathrm{a}_{\mathrm{i}^L}^L + s_{\mathrm{i}^L} + t_{\mathrm{i}^L j}\}; \tag{2.150}$$

$$\mathrm{q}^{L'} = \begin{cases} \mathrm{q}^L + q_j & \text{if } j \in P' \\ \mathrm{q}^L - q_j & \text{if } j \in D' \end{cases} \tag{2.151}$$

$$\mathrm{r}^{L'} = r; \tag{2.152}$$

$$\mathrm{c}^{L'} = \mathrm{c}^L + c_{\mathrm{i}^L j}; \tag{2.153}$$

$$\mathcal{V}^{L'} = \begin{cases} \mathcal{V}^L & \text{if } j \in P' \\ \mathcal{V}^L \cup \{r\} & \text{if } j \in D' \end{cases} \tag{2.154}$$

$$\mathcal{O}^{L'} = \begin{cases} \mathcal{O}^L \cup \{r\} & \text{if } j \in P' \\ \mathcal{O}^L \setminus \{r\} & \text{if } j \in D' \end{cases} \tag{2.155}$$

$$\mathrm{p}^{L'} = L; \tag{2.156}$$

The above discussion omits to consider the true feasibility test for ride time constraints. Condition (2.147) is only a necessary but not sufficient condition for the ride time feasibility: initializing the forward labeling approach in such a way that the depot departure time is as early as possible ($t_{o^+} = e_{o^+}$), results in the earliest possible start service times for the nodes of the path obtained through dynamic programming approach. Such early start service times can result in the violation of the ride time constraint for some requests in the process of forward labeling algorithm, while there might exist a different value for start service times, for which the ride time constraint is not violated.

Let us consider the heuristic way in which we handle the ride time issue by computing the maximum feasible delay at the pickup node of requests whose ride time is violated. Such feasible delay is called the "forward time slack", which is first proposed by Cordeau [12]. To compute the forward time slack, first we need to find a sub-path of the current partial path.Let $\mathcal{P}$

denotes such sub-path; $\mathcal{P}$ must have some characteristics:

- the beginning of the sub-path is a pickup node.
- the sub-path must include the pickup node of a request whose ride time is violated.
- the cardinality of the set of open request must be one at the beginning of the sub-path.

The forward time slack of the sub-path $\mathcal{P}$ is then as follows:

$$\mathcal{F}_{\mathcal{P}} = \min_{\bar{p} \leq i \leq \mathrm{i}^L} \{\mathrm{a}_i^L - (\mathrm{a}_{\bar{p}}^L + \sum_{\bar{p} \leq k < i} (s_k + t_{k,k+1})) + \min\{\bar{l}_i - \mathrm{a}_i^L, b_r^f - h_r\}\} \quad (2.157)$$

where $\bar{p}$ is the beginning of the sub-path and h_r is the value of the ride time calculated as follows:

$$h_r = \begin{cases} \infty & \text{if } i \in P' \\ \mathrm{a}_{d_r}^L - \mathrm{a}_{p_r}^L - s_{p_r} & \text{if } i \in D' \end{cases} \quad (2.158)$$

After obtaining the feasible delay, the elements of $\mathcal{A}^L$ are updated along the sub-path $\mathcal{P}$, accordingly. At the beginning of the sub-path we have:

$$\mathrm{a}_{\bar{p}}^L = \mathrm{a}_{\bar{p}}^L + \mathcal{F}_{\mathcal{P}} \quad (2.159)$$

This can be interpreted as waiting time for the vehicle at $\bar{p}$. Then, for each node $i \prec i'$ along the sub-path we have:

$$\mathrm{a}_{i'}^L = \max\{e_{i'}, \mathrm{a}_i^L + s_i + t_{ii'}\} \quad (2.160)$$

After that, the ride time constraint is verified once again using the updated start service times. Thus, the ride time feasibility issue is handled here by finding a feasible way to delay some start service time to meet the maximum ride time value. The partial path is then extended with those delays being fixed. We insist on the fact that this is only a heuristic method to handle the ride time constraint, as $\mathcal{F}_{\mathcal{P}}$ yields the maximum value for the feasible delay; therefore, the current dynamic programming approach as a whole become a heuristic method when the ride time constraint become tight, under Assumption 6-a.

Once a new label L' is created, it is possible to apply the concept of dominance rule. An exact dominance rule is the one already discussed. The dominance rule that we have applied for the numerical experiments are as follows, which are not exact:

Dom"1:

$$\begin{aligned}
\mathrm{i}^{L'} &= \mathrm{i}^{L''} && (2.161)\\
\mathrm{a}^{L'}_{\mathrm{i}^{L'}} &\leq \mathrm{a}^{L''}_{\mathrm{i}^{L''}} && (2.162)\\
\mathrm{c}^{L'} &\leq \mathrm{c}^{L''} && (2.163)\\
\mathcal{V}^{L'} &\subseteq \mathcal{V}^{L''} && (2.164)\\
|\mathcal{O}^{L'}| = |\mathcal{O}^{L''}| &= \begin{cases} 1 & \text{if } \mathrm{i}^{L'} \in P' \\ 0 & \text{if } \mathrm{i}^{L'} \in D' \end{cases} && (2.165)
\end{aligned}$$

Dom"2:

$$\begin{aligned}
\mathrm{i}^{L'} &= \mathrm{i}^{L''} && (2.166)\\
\mathrm{a}^{L'}_{\mathrm{i}^{L'}} &= \mathrm{a}^{L''}_{\mathrm{i}^{L''}} && (2.167)\\
\mathrm{c}^{L'} &\leq \mathrm{c}^{L''} && (2.168)\\
\mathcal{V}^{L'} &\subseteq \mathcal{V}^{L''} && (2.169)\\
|\mathcal{O}^{L'}| = |\mathcal{O}^{L''}| &= \begin{cases} 1 & \text{if } \mathrm{i}^{L'} \in P' \\ 0 & \text{if } \mathrm{i}^{L'} \in D' \end{cases} && (2.170)
\end{aligned}$$

2.6. Numerical Experiments

2.6.1 Objectives of the Experiments

We proceed to compare numerically the different variants of the labeling algorithms proposed earlier in sections 2.3 and 2.5 for the one-vehicle problem of the mathematical formulations presented in the first chapter (with and without the request consolidation assumption). Our goal is to evaluate the impact of:

1. Different variants of dominance rule, which were presented along the text. We are comparing in our experiments Dom1 and Dom2 for the problem of section 2.3 and Dom"1 and Dom"2 for the problem of section 2.5.

2. The size of the state space depending on whether we insist on generating elementary paths or an ng-paths.

We expect that applying Dom2 (similarly Dom"2) in one direction results in an algorithm (Table 2.3) which is computationally faster than the algorithm of Table 2.2 where Dom1 (similarly Dom"1) is used in two directions. Besides, we expect that generating ng-path results in a faster algorithm than the elementary one.

2.6.2 Experimental Scheme

We outline different versions of the procedure used in our experiments. Below, the versions are presented for the elementary shortest path problem with time windows, pickup and delivery, load, and ride time constraints discussed in section 2.5. The elementary shortest path problem with time windows discussed in section 2.3 can be seen as a special case where Dom"1(or Dom"2) is replaced by Dom1 (or Dom2).

1. VersionA: apply Dom"1 in two directions, while elementary paths are generated.

2. VersionB: apply Dom"1 in two directions, while ng-paths are generated.

3. VersionC: apply Dom"1 in one direction, while elementary paths are generated.

4. VersionD: apply Dom"1 in one direction, while ng-paths are generated.

5. VersionE: apply Dom"2 in two directions, while elementary paths are generated.

6. VersionF: apply Dom"2 in two directions, while ng-paths are generated.

7. VersionG: apply Dom"2 in one direction, while elementary paths are generated.

8. VersionH: apply Dom"2 in one direction, while ng-paths are generated.

9. VersionI: apply the algorithm proposed by Robke and Cordeau [38].

Implementations are done in the environment of BaPCod, which is a prototype of a generic branch-and-price approach [46]. We apply a column generation algorithm at the root node of the branch-and-price tree and at each iteration we solve the pricing subproblem with each of the above versions. As each of the above versions can yield a different solution (even though its cost is the same), we return the solution of VersionA systematically (so we can reproduce the run). For the comparison, we record:

- The number of labels that are created to generate a specific solution using different versions.
- The number of labels that are either eliminated or dominated using different versions.
- Computational time of each version.

2.6.3 Computational Results

A. Experimental Setting

The experiments are implemented in C++ , under the environment of BaPCod and run on an MacBookPro5,2 (3,06 GHz) over a data set which is obtained from the generated data available on

http://www.hec.ca/chairedistributique/data/darp.

Tables 2.5 and 2.6 outline the average of the desired quantities on the whole set of instances for each version. For both tables, the CPU time is outlined in the column entitled **T**. The unit of time is second. The total number of created labels, eliminated labels, and dominated labels are presented in columns **CL**, **EL**, and **DL**, respectively. Eliminated labels refer to the number of labels that are not created because of not being able to reach o^-. Note that here the size of ng-sets is 3. The computations are done with a two-digit precision.

B. Analysis of the Experiments

Considering the obtained results, VersionD of the algorithm is computationally more efficient. As it was predicted generating, ng-paths improves the performance considerably. Applying dominance rule in one direction can

Table 2.5: Average of the results for the case where request consolidation is assumed.

	T	CL	EL	DL
VersionA	216.44	333083.69	8554.08	154322.08
VersionB	0.61	98637.54	9618.54	452471.77
VersionC	117.25	331743.62	17093.92	291361.31
VersionD	0.21	98637.54	9618.54	225346.15
VersionE	211.19	619553.85	28082.92	408674.61
VersionF	3.82	334518.46	19894.46	77446.15
VersionG	136.47	619553.85	28082.92	389371.54
VersionH	1.49	334518.46	19894.46	337705.38
VersionI	0.20	6239.38	8546.62	156521.54

be assumed as second effective parameter. For the first case (under the request consolidation assumption) the number of dominated labels is greater than the number of eliminated ones. For the majority of the instances of the second case (without the request consolidation assumption), the number of eliminated labels is greater than dominated one; as both procedures eliminate considerable number of labels, both of them are recommended to be applied in the process of the forward labeling algorithm. Note that the algorithm proposed by Robke and Cordeau [38] for the pickup and delivery problem with time windows is efficient for the first case (under the request consolidation assumption), while it is not efficient for the second case. In fact, for the second case (without the request consolidation assumption), all the versions of our algorithm (but VersionE) are computationally better than VersionI. The structure of our algorithm seems to be more efficient. In their algorithm, they add all the created labels to $\mathcal{L}$. When a pop is done, the algorithm checks if the poped label can be dominated by already existing labels at that node. If it is dominated, feasibility arc extensions are not verified; otherwise, the feasibility conditions are tested to extend the poped label. Finally, Figures 2.5 to 2.10 are provided to make the comparison of the results easier.

Conclusion

In this chapter, different variants of resource constraint shortest path problem are studied, together with the forward labeling algorithm as their solution approach. The computational complexity of different variants of forward labeling algorithm is then discussed. Finally, the efficiency of different versions of the algorithms is evaluated numerically. Considering the

Table 2.6: Average of the results for the case where request consolidation is not assumed.

	T	CL	EL	DL
VersionA	382.10	1318170	405167	365582
VersionB	24.51	978572	307149	1207090
VersionC	178.35	1318170	405167	259637
VersionD	7.26	978572	307149	295300
VersionE	411.81	1981870	589226	549018
VersionF	33.68	1312470	398185	1498150
VersionG	224.56	1981870	589226	432813
VersionH	10.11	1312470	398185	414541
VersionI	405.83	2469030	673640	529651

experimental results, applying state-space relaxation together with partial dominance rule improves the performance of the computations considerably. Moreover, it is expected that applying VersionD of the algorithm can improve on the results obtained for the pickup and delivery problem with time windows by Robke and Cordeau [38].

Figure 2.5: Total number of created labels for the case where request consolidation is assumed.

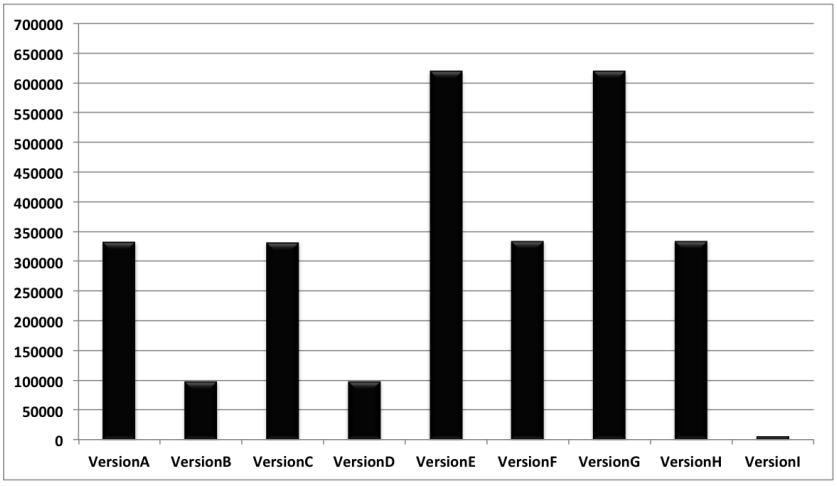


Figure 2.6: Total number of created labels for the case where request consolidation is not assumed.

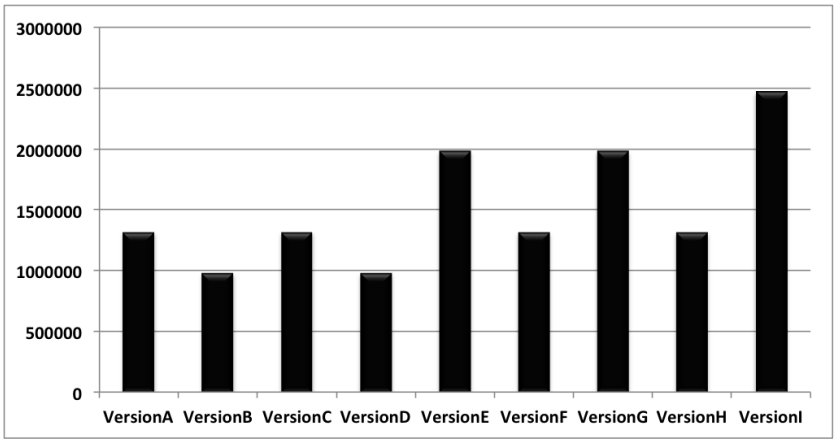


Figure 2.7: Total number of eliminated labels for the case where request consolidation is assumed.

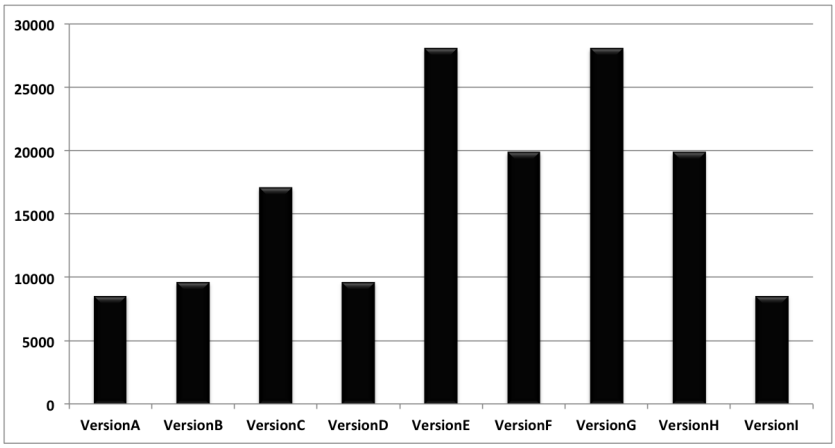


Figure 2.8: Total number of eliminated labels for the case where request consolidation is not assumed.

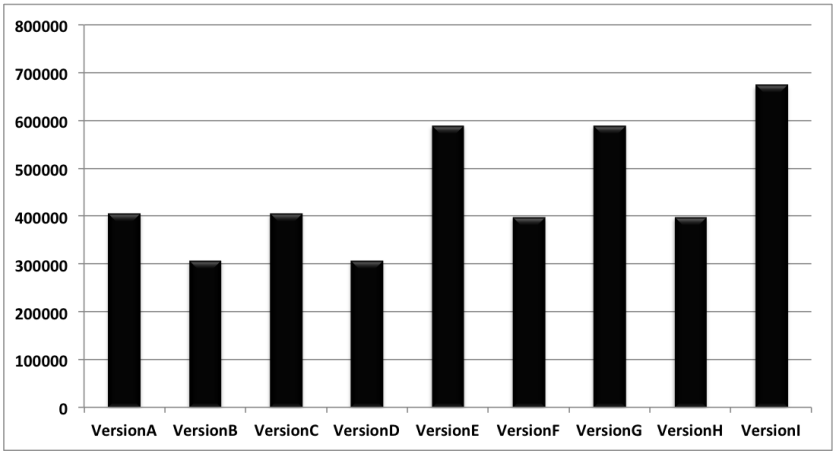


Figure 2.9: Total number of dominated labels for the case where request consolidation is assumed.

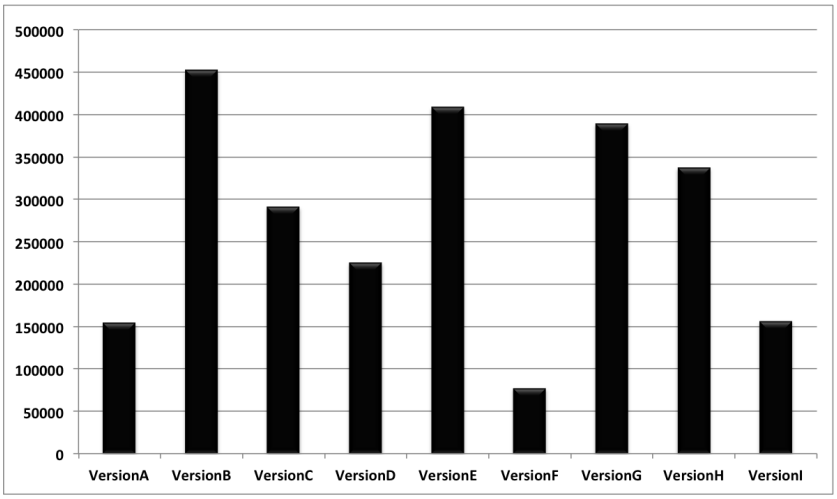
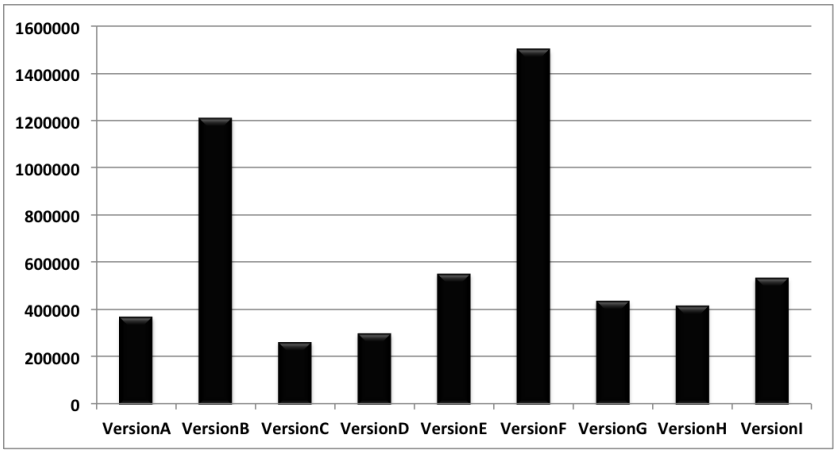


Figure 2.10: Total number of dominated labels for the case where request consolidation is not assumed.



3

Solving the Static Model

Introduction

The linear relaxation of original compact formulations presented in sections 1.4.1 and 1.4.2 are very weak to be used in a branch-and-bound approach. In contrast, the branch-and-price approach that relies on a Dantzig-Wolfe reformulation provides tighter dual bounds usually. In this chapter, we are interested to apply a branch-and-price approach to solve the Dantzig-Wolfe reformulation of mixed-integer linear programs presented in sections 1.4.1 and 1.4.2. First we outline the Dantzig-Wolfe reformulations of the compact formulations. Then, the obtained reformulations are solved using a branch-and-price approach. Some aspect of the methodology is discussed and experimentally tested using the environment of BaPCod[1].

3.1. The Dantzig-Wolfe Reformulation

Many mixed-integer linear programs have a decomposable structure, which makes them well suited for the Dantzig-Wolfe reformulation and for which branch-and-price can be a competitive solution approach [28]. The dial-a-ride problem is not an exception. The Dantzig-Wolfe reformulation of the compact formulations presented in sections 1.4.1 and 1.4.2 are as follows:

[1]BaPCod is a prototype code that solves Mixed Integer Programs (MIP) using a branch-and-price algorithm on a Dantzig-Wolfe reformulation.

$$\min \quad \sum_{i=0}^{m-1} \sum_{g_i \in Z_i} c_{g_i} \nu_{g_i} \tag{3.1}$$

$$\text{s.t}$$

$$\sum_{i=0}^{m-1} \sum_{g_i \in Z_i} a_{pg_i} \nu_{g_i} = 1 \quad \forall p \in P' \tag{3.2}$$

$$\sum_{g_i \in Z_i} \nu_{g_i} = n_i \quad i = 0, ..., m-1 \tag{3.3}$$

$$\nu_{g_i} \in \mathbb{Z}_+ \tag{3.4}$$

where Z_i is the subproblem polyhedron that contains all feasible routes g_i satisfying constraints (1.6) to (1.20) of the compact formulation presented in section 1.4.1 (for the mathematical formulation of section 1.4.2, constraints (1.25) to (1.33) generate Z_i). As it is mentioned in Chapter 1, there exist m types of vehicles and $n_i, i \in \{0, 1, ..., m-1\}$, vehicles of each type. Thus, for each type i, there exist n_i identical subproblems Z_i. This means that the pricing problem involves non-identical subproblems (one for each vehicle type), each of which decomposes into identical subsystems, one for each vehicle of that type. As we have identical subproblems, symmetry is possible. To avoid this, the Dantzig-Wolfe reformulation is defined based on the aggregate variable $\nu_{g_i} = \sum_{j=1}^{n_i} \lambda_{g_i}^j$, where $\lambda_{g_i}^j = 1$, if vehicle j of type i generates route g_i; otherwise, $\lambda_{g_i}^j = 0$. Thus, ν_{g_i} represents the number of times a route g_i is chosen in the solution. Finally, c_{g_i} is the cost of the generated route g_i, and a_{pg_i} is one if route g_i contains node p; if not, $a_{pg_i} = 0$.

The LP solution to the Dantzig-Wolfe reformulation (3.1) to (3.4) provides a dual bound that is typically tighter than that of the LP relaxation of the original compact formulation and is equal to the best bound that can be derived by Lagrangian relaxation of the request covering constraints that are viewed as hard constraints here. However, such LP relaxation encompass a large number of variables, as any increase in the number of requests to be covered, results in exponential increase in the size of Z_i. To manage this drawback, a column generation technique is applied. Finally, the column generation approach is embedded in a branch-and-bound tree to guarantee the integrality of the solution. The overall scheme is known as branch-and-price approach.

3.2. The Branch-and-Price

The LP relaxation of the Dantzig-Wolfe reformulation (3.1) to (3.4) is traditionally called master problem (MP), which has a very large number of variables. To apply the column generation approach to solve MP, a restricted master linear program (RMLP) is generated by considering a subset of feasible routes $\bar{Z}_i \subset Z_i$:

$$(\text{RMLP}) : \min \quad \sum_{i=0}^{m-1} \sum_{g_i \in \bar{Z}_i} c_{g_i} \nu_{g_i} \tag{3.5}$$

$$\text{s.t}$$

$$\sum_{i=0}^{m-1} \sum_{g_i \in \bar{Z}_i} a_{pg_i} \nu_{g_i} = 1 \quad \forall p \in P' \tag{3.6}$$

$$\sum_{g_i \in \bar{Z}_i} \nu_{g_i} = n_i \quad i = 0, ..., m-1 \tag{3.7}$$

$$\nu_{g_i} \in \mathbb{Z}_+ \tag{3.8}$$

whose dual formulation is

$$(\text{DRMLP}) : \max \quad \sum_{p \in P'} \omega_p + \sum_{i=0}^{m-1} \sigma_i \tag{3.9}$$

$$\text{s.t}$$

$$\sum_{p \in P'} a_{pg_i} \omega_p + \sigma_i \leq c_{g_i} \quad \forall i = 0, ..., m-1, g_i \in \bar{Z}_i \tag{3.10}$$

$$\omega_p \in \mathbb{R} \tag{3.11}$$

$$\sigma_i \in \mathbb{R} \tag{3.12}$$

where ω_p and σ_i are the dual variables corresponding to constraints (3.6) and (3.7), respectively. Note that considering $\geq$ for relation (3.6) and $\leq$ for relation (3.7) will speed up the convergence of column generation, as $\omega_p \geq 0$ and $\sigma_i \leq 0$.

Let $\nu = (\nu_{g_i})_{\{g_i \in \bar{Z}_i : i=0,...,m-1\}}$ represent the solution to the primal problem obtained by applying the simplex algorithm on RMLP and $\omega = (\omega_p)_{p \in P'}$ and $\sigma = (\sigma_i)_{\{i=0,...,m-1\}}$ denote the corresponding dual solutions. Suppose that

(ω^*, σ^*) is the optimal solution of DRMLP associated to primal solution ν^*. If (ω^*, σ^*) is feasible over the original dual space, then ν^* and (ω^*, σ^*) are optimal. If not, there exists a route in $Z_i \setminus \bar{Z}_i$ that violates constraint (3.10). To find that route we need to call the forward labeling approach explained in section 2.5 (and 2.3 for model of section 1.4.2), whose objective function is:

$$c_{g_i} - \sum_{p \in P'} a_{pg_i} \omega_p^* - \sigma_i^* \leq 0 \tag{3.13}$$

In this way $\bar{Z}_i$ is gradually extended in such a way that it includes all the feasible routes that participate in the optimal solution. For more details refer to [18].

Since the simplex algorithm is applied to solve RMLP, a feasible primal solution is needed to initialize it. Such primal solution can be provided either using a heuristic approach and/or introducing artificial columns in the RMLP. A proper initialization may reduce *heading-in effect*, one drawback of column generation strategy, by providing more information to produce tighter dual bounds and more relevant columns at the beginning of the procedure.

3.2.1 Tradeoff between Dual Bound Quality and Computing Time of the Column Generation Procedure

The quality of the dual bound obtained by the column generation approach depends on the selected model for the pricing problem. A richer pricing model that captures more of the combinatorial structure of the problem shall lead to better dual bounds but higher computing times spent in the pricing oracle. The best dual bound is achieved by generating elementary paths in the subproblem. It is called the elementary bound. However, obtaining such quality dual bound is computationally less efficient in practice. On the contrary, relaxing the elementarity usually results in lower quality dual bounds, but in more reasonable amount of computing time. An alternative is to generate partially elementary paths. One way to generate such kind of paths is to generate ng-routes discussed in Chapter 2.

In 2013, Contardo et al. [11] introduced a family of valid inequalities called Strong Degree Constraints (SDCs). For each request $r \in \mathcal{R}$ and route

$g_i \in Z_i$, let $\zeta_r^{g_i}$ be one if route g_i visits request r; otherwise, $\zeta_r^{g_i} = 0$. The Strong Degree Cut is then as follows:

$$\sum_{g_i \in Z_i} \zeta_r^{g_i} \nu_{g_i} \geq 1 \tag{3.14}$$

If the procedure applied to the pricing problem allows the generation of non-elementary routes and constraint (3.2) relaxes to $\geq$, then adding SDCs to the master problem can eliminate a solution where a fraction of a route which cycles at a node is used to cover that node. In other words, partial elementarity of the routes is imposed by non-robust SDCs. These constraints are non-robust, as the value of the corresponding dual variables cannot be translated into subproblem costs. Hence, the forward labeling algorithm applied for the pricing problem needs some adaptation to manage the effect of these non-robust cuts. Doing so, the dimension of the label is extended per cut. Such label dimension is only used to determine if the value of the dual variable associated to a cut should be subtracted from the cost of the new label or not. This can be interpreted as rewarding/penalizing certain extensions. For more details refer to [45].

Let RMLP be enhanced with the SDCs associated to each request $r \in \mathcal{R}$ as follows:

$$\sum_{g_i \in \bar{Z}_i} \zeta_r^{g_i} \nu_{g_i} \geq 1 \quad \forall r \in \mathcal{R} \tag{3.15}$$

Suppose that π_r is the corresponding dual variable. The aim is to outline how the forward labeling algorithm is adapted to take into account the effect of π_r, when label L is extended to a pickup node p_r. If $\pi_r > 0$, its effect must be taken into account by subtracting the value of π_r from the cost of the new label L' at p_r if $r \notin \mathcal{V}^L \cup \mathcal{O}^L$, (i. e. $c^{L'} - \pi_r$). Since $\pi_r > 0$, this particular extension is being rewarded.
The dominance rule in the definition of forward labeling algorithm also needs to be modified as follows:

$$\begin{aligned}
\mathrm{i}^{L} &= \mathrm{i}^{L'} && (3.16)\\
\mathrm{a}_j^{L} &\leq \mathrm{a}_j^{L'} && (3.17)\\
\mathrm{c}^{L} &\leq \mathrm{c}^{L'} - \sum\nolimits_r \pi_r \quad r \notin \mathcal{V}^{L'}, r \in \mathcal{V}^{L} && (3.18)\\
\mathcal{O}^{L} &= \mathcal{O}^{L'} && (3.19)\\
|\mathcal{O}^{L}| = |\mathcal{O}^{L'}| &= \begin{cases} 1 & \text{if } \mathrm{i}^{L} \in P' \\ 0 & \text{o.w} \end{cases} && (3.20)
\end{aligned}$$

The traditional definition of resources for not visiting a node more than once using $\mathcal{V}^L$ indicates that label L, whose partial path visits any vertex not visited by the partial path at L', can never dominate label L'. In contrast, when SDCs are applied to impose elementarity, label L can dominate label L' if c^L is sufficiently smaller than $\mathrm{c}^{L'}$ to compensate the dual variables corresponding to the vertices not in $\mathcal{V}^{L'}$. This results in discarding more labels than the traditional dominance rule defined for forward labeling algorithms. All in all, applying SDCs are not only an alternative to reach dual bounds but also they decrease the time required to solve the subproblem.

Another alternative to accelerate the column generation procedure is to adopt a stage-strategy to solve the pricing problem. At each stage, one solves the subproblem with more and more accurate heuristics in comparison to former stages. At the last stage an exact solver is used. What we are using is a two stage approach. The heuristic applied to solve the pricing problem in the first stage is obtained through relaxing time consuming criteria of traditional dominance rule:

$$\begin{aligned}
\mathcal{V}^{L'} &\subseteq \mathcal{V}^{L} && (3.21)\\
\mathcal{O}^{L'} &= \mathcal{O}^{L} && (3.22)
\end{aligned}$$

together with the calculations of forward time slack, that is needed when a ride time constraint is violated. The column generation approach calls the first stage as long as it returns a route with negative reduced cost. After that, the exact routine of the second stage is called. The procedure that we have applied not only helps to eliminate time consuming criteria of dominance rule, but also it allows to eliminate more labels.

3.2.2 The Branching Strategy

On completion of the column generation procedure applied to MP, the integrality of the obtained solution is checked. If it is not integer, the factional solution is eliminated by a disjunctive branching constraint. The design of branching scheme must be done in such a way that it not only leads to the integrality of the solution, but also yields dual bound improvements. When a decomposition approach is used, the following issues must also be considered for the choice of the branching scheme:

- To decide branching either on variables of the original compact formulation, or those of the Dantzig-Wolfe reformulation, or branching on constraints.
- The place where to enforce the branching constraint: either in the master or in the subproblem.
- The compatibility of branching decisions with the structure of the pricing problem.

To the best of our knowledge, a branch-and-price approach has not been applied to the dial-a-ride problem. The work of Parragh et al. [31] is the sole contribution that solved heterogenous DARP at the route node of the branch-and-price tree. The branching scheme that we use is the automated scheme of BaPCod. In the sequel, we provide a brief review to outline how the branching schemes available on the BaPCod work. The presented material herein is based on the work of Vanderbeck [47].

As branching directly on fractional $\nu^*_{g_i}$ is rather weak, the first scheme of BaPCod is to branch on a combination of $\nu^*_{g_i}$. One way to find such combination is to map ν^* into the original compact formulation variables. Let X^{g_i} be the route with respect to $\nu^*_{g_i}$. Assume that $X^{g_i}_l$ is the component l of the route X^{g_i}. Let $\sum_{g_i \in \bar{Z}_i} X^{g_i}_l \nu_{g_i}$ takes a fractional value α. The disjunctive branching constraints are then as follows:

$$\sum_{g_i \in \bar{Z}_i} X^{g_i}_l \nu_{g_i} \leq \lfloor \alpha \rfloor \tag{3.23}$$

$$\sum_{g_i \in \bar{Z}_i} X^{g_i}_l \nu_{g_i} \geq \lceil \alpha \rceil \tag{3.24}$$

If component l is corresponded to the arc variables x_{ij}, then the above branching scheme can be interpreted as branching on the flow of arc (i, j).

As we have identical subproblems, branching constraints on aggregate variables are typically not sufficient to eliminate all fractional solutions. Hence, the second level of branching scheme of BaPCod can be called. The second class of branching schemes can be enforced directly into the subproblem. Let $\nu^* = (\nu^*_{g_i})_{\{g_i \in \bar{Z}_i : i=0,...,m-1\}}$ be a master solution. A geometrical view of ν^* is to assume m (number of type of vehicles) rectangles each width of which is $n_i, (i = 0, ..., m-1)$. For rectangle i, each column, $g_i \in \bar{Z}_i$, defines a strip of width $\nu^*_{g_i}$ at a specific position. The goal of the branching scheme is to partitioning the rectangle of width n_i into n_i sub-strips of width 1, each of which shall be associated with an index $j = 1, ..., n_i$.

During the procedure, branching constraints recursively partition the subproblem solution set. Let $\bar{Z}_i$ be the set of current generated columns sorted in lexicographical order. The scheme first seeks for a component bound sequence S, which is an ordered set of bounding restrictions on the components of the subproblem solution vector X^{g_i}. Then, it looks for a subset $\hat{Z}_i \subseteq \bar{Z}_i$ such that the members of $\hat{Z}_i$ satisfy the component bounds defined via S and $\sum_{g \in \hat{Z}_i} \nu^*_{g_i} = \alpha \notin \mathbb{Z}$. Then, the two branches of the up-branch are

$$\sum_{g_i \in \hat{Z}_i} \nu^*_{g_i} \geq \lfloor \alpha \rfloor \tag{3.25}$$

$$\sum_{g_i \in Z_i \setminus \hat{Z}_i} \nu^*_{g_i} \geq n_i - \lceil \alpha \rceil \tag{3.26}$$

This latter scheme can be interpreted as branching on the fleet size. This particular implementation permits the use of the original pricing problem oracle after branching if the pricing oracle can handle bounds on the subproblem variables. Since branching constraints are not dualized, they induce better improvements of dual bounds. For more details refer to Vanderbeck [47].

In addition to the generic branching schemes provided by Vanderbeck [47], there exists another scheme, which is more application specific. In 1995, Gélinas et al. [20] proposed another scheme to impose branching decisions into

the pricing problem. Their approach is for vehicle routing problems with time windows, where the travel time and the service time are both integer. The idea is to branch on subproblem variables. More precisely, branching decisions are made on resource variables (e.g., time or capacity) rather than network flow variables. Their computational experiments showed that time-based branching decreases the number of nodes explored by two thirds and the total computational time by more than half compared to flow based branching.

This scheme consists of splitting the time window of a node and creating a branch for each subinterval. A candidate node for branching is the one for which the time window can be divided into two intervals so that at each branch, at least one route becomes infeasible. In our numerical experiment, we apply the branching strategies of [47] only and hence we cannot provide any comparison with the scheme of Gélinas et al. [20].

3.3. Numerical Experiments

3.3.1 Objectives of the Experiments

Our aim is to solve the Dantzig-Wolfe reformulation presented in section 3.1 using a branch-and-price approach and compare the results with that of the branch-and-cut approach of Cordeau [12].

Applying the "request consolidation" assumption presented in Chapter 1 results in relaxing the ride time constraint, the precedence constraint, and the load constraint. Hence, we expect that solving the Dantzig-Wolfe reformulation of the original compact formulation under such assumption (presented in section 1.4.2) is computationally more efficient. It provides an upper bound on the optimal solution of the generic compact formulation where request consolidation is not assumed.

We are also interested to apply different accelerating techniques and compare their effect. One technique is to apply the solution of the model under request consolidation assumption as an initialization for the column generation approach. As VersionD of the forward labeling approach is used to

solve the pricing problem, it is possible to have some non-elementary dual bounds. Although time windows can help to avoid the generation of such non-elementary paths, we use strong degree cuts to reach the elementary bounds. We expect that applying strong degree cuts not only leads to elementary bounds but also reduce the computational time. We then apply a two stage approach to solve the subproblem. Applying such approach is also expected to reduce the computational times. Finally, a branching scheme is applied on master variables to solve the problem to integrality.

3.3.2 Experimental Scheme

At each iteration of the column generation the master problem is solved using the simplex algorithm. To solve the pricing problem different schemes are used:

- **S1 :** The branch-and-price is applied at the root node, and the pricing problem is solved using a MIP solver.
- **S2 :** The branch-and-price is applied at the root node, and the pricing problem is solved using VersionD of the forward labeling algorithm explained in Chapter 2.
- **S3 :** To accelerate the method, we initialize the simplex algorithm using the solution of the model under request consolidation assumption, while the branch-and-price approach is applied at the root node and the pricing problem is solved using VersionD of the forward labeling algorithm.
- **S4 :** As VersionD of the forward labeling algorithm generates ng-routes, we apply the strong degree cuts not only to reach to elementary bounds but also to accelerate the method.
- **S5 :** Another procedure that speeds up the solving of the pricing problem at the root node, is a two-stage scheme: in the first stage we apply a dominance rule that relaxes the following two criteria:

$$\mathcal{O}^{L'} = \mathcal{O}^{L''}$$
$$\mathcal{V}^{L'} \subseteq \mathcal{V}^{L''}$$

Besides, the re-calculation of the ride time using forward time slack is not performed. Then, for the second stage the MIP solver is used.

Finally, for the best scheme we apply the generic branching scheme of [47] to guarantee the integrality of the solution; **S'i** refers to the case where scheme **Si** together with branching upto 1000 node is applied.

3.3.3 Computational Results

The experiments are implemented in C++ , under the environment of BaPCod and run on an MacBookPro5,2 (3,06 GHz) over the data set available on

http://www.hec.ca/chairedistributique/data/darp.

In all tables, DB represents the value of the dual bound and if it is in bold, it indicates the optimal value. T_m and T_{sp} refer to the total time needed to solve the master problem and the subproblem, respectively. The reported times are in second. In all tables, if any value is marked by *, it means that the computational time to obtain that solution is smaller than the one obtained via the branch-and-cut algorithm of Cordeau [12]. For all experiments, the time limit of one hour is considered, while Cordeau [12] assumed four hours time limit for his experiments. If the algorithm is stopped because of the time limit, it is reported by T.

3.3.4 Analysis of the Experiments

Table 3.1 outlines the result of applying the branch-and-cut approach of Cordeau [12]. These results are reported directly from their paper. Table 3.2 represents the solution of the model where request consolidation is assumed. Because the number of the available vehicles are limited, such assumption can make the problem infeasible, as it can be seen for instances (a3-30) and (b2-20).

As it is reported in Table 3.3, scheme S1 can solve 70.83% of instances, among which 52.94% of them are solved to optimality. Scheme S2 could solve 37.5% of instances in the time limit of one hour. The re-computing of

the ride time along the partial paths generated in the process of the forward labeling algorithm together with the applied dominance rule, which is weak because of the existence of the ride time constraint, are two drawbacks that cause such increase in the computational times, when scheme S2 is applied. However, the computational time for the instances that are solved using S2 is smaller than the case where S1 is applied. Besides, instances (a4-16) and (b4-16) are solved significantly faster than the existing benchmark.

It was predicted that initializing the simplex algorithm using the solution obtained through scheme S'2 can help to improve the performance of the methodology. Considering the results reported in Table 3.4, such a thing is valid on average; furthermore, the initialization results in solving one more instance. Since no non-elementary bound is yielded using scheme S2, applying strong degree cuts did not result in adding any cuts into the master problem; anyhow, adjusting the forward labeling approach based on this idea improved the efficiency of the method in such a way that 66.67% of instances are solved, and five instances are solved computationally faster than their counterparts that are solved using the branch-and-cut approach.

It was expected that applying S5 can improve the efficiency of the method. On average, such a result is obtained comparing S1. However, this is not the case for all instances. For some instances, the same dual bound is obtained in a greater amount of time comparing to the case where S1 is applied. This can be justified by the fact that relaxing the demanding criteria of dominance rule together with not re-calculating the ride time constraint does not necessarily result in an efficient heuristic. Finally, applying the generic branching scheme helped to solve one more instances to optimality in a smaller amount of time comparing the benchmark.

Conclusion

In this chapter, some aspects of the branch-and-price approach that relies on the Dantzig-Wolfe reformulation of the compact formulations presented for the dial-a-ride problem in sections 1.4.1 and 1.4.2 are studied. This is the first work that solves the dial-a-ride problem using a branch-and-price approach, while ride time constraints are considered directly in the sub-problem. As

it was predicted, considering request consolidation assumption reduces the problem to the standard vehicle routing problem with time windows, which is less difficult than the original DARP. For this case, all the instances are solved to optimality in a reasonable amount of time (however this only approximates the solution of the original problem). Besides, such assumption can cause infeasibility due to the limited number of available vehicles.

On the other hand, five different schemes are applied to solve the Dantzig-Wolfe reformulation of the generic compact formulation where no specific assumption is considered. All the schemes are theoretically discussed and numerically experimented. According to the numerical results, scheme S4, which refers to applying strong degree cuts, not only outperforms the other schemes but also it has solved six instances computationally better than the existing benchmark in the literature that are obtained by applying a branch-and-cut approach.

Table 3.1: Results of the branch-and-cut approach proposed by Cordeau [12]

Instance	DB	T_m	Node	Cuts
a2-16	**294.25**	0.6	4	46
a2-20	**344.83**	4.8	181	93
a2-24	**431.12**	16.2	444	140
a3-18	**300.48**	12.0	565	172
a3-24	**344.83**	250.2	6.863	324
a3-30	**494.85**	2543.4	43632	425
a3-36	**583.19**	747.6	7451	423
a4-16	**282.68**	152.4	6615	404
a4-24	**375.02**	1511.4	28939	351
a4-32	447.66	14400.0	105500	806
a4-40	475.05	14400.0	32200	716
a4-48	486.03	14400.0	21400	1019
b2-16	**309.41**	9.0	487	130
b2-20	**332.64**	0.6	2	30
b2-24	**444.71**	7.8	146	103
b3-18	**301.64**	42.0	2458	245
b3-24	**394.51**	217.2	8147	252
b3-30	**531.44**	409.2	9862	274
b3-36	**603.79**	3724.2	66079	291
b4-16	**296.96**	47.4	2579	196
b4-24	**371.41**	351.0	7119	255
b4-32	**494.82**	10609.2	126332	375
b4-40	591.76	14400.0	81100	632
b4-48	586.91	14400.0	21000	753
Ave.		77.78		

Table 3.2: Comparison of applying S'1 and S'2 on the Dantzig-Wolfe reformulation of the compact formulation presented in section 1.4.2 under the Consolidation Assumption

	S'1			S'2		
Instance	DB	T_m	T_{sp}	DB	T_m	T_{sp}
a2-16	**372.57**	3.52	3.38	**372.57**	0.24	0.15
a2-20	**446.56**	12.47	12.15	**446.56**	1.00	0.78
a2-24	**542.57**	24.87	24	**542.57**	2.63	2.35
a3-18	**380.23**	3.97	3.80	**380.23**	0.33	0.25
a3-24	**434.29**	16.16	15.60	**434.29**	1.44	1.19
a3-30	-	I	-	-	I	-
a3-36	**786.31**	420.81	417.63	**786.31**	19.18	17.79
a4-16	**365.68**	4.44	4.33	**365.68**	0.13	0.09
a4-24	**540.53**	26.76	26.09	**540.53**	1.59	1.13
a4-32	**678.16**	174.02	170.73	**678.16**	10.78	8.77
a4-40	**865.11**	571.50	568.10	**865.11**	30.36	27.41
a4-48	-	T	-	**990.29**	229.78	193.91
b2-16	**348.84**	10.21	9.79	**348.84**	1.09	0.83
b2-20	-	I	-	-	I	-
b2-24	**517.67**	89.65	87.48	**517.67**	10.70	8.97
b3-18	**378.56**	7.93	7.60	**378.56**	0.52	0.36
b3-24	472.29[1]	25.23	24.56	**465.89**	2.87	2.12
b3-30	**643.09**	85.84	84.34	**643.09**	5.27	4.58
b3-36	**733.13**	126.38	123.77	**733.13**	30.09	27.77
b4-16	**381.30**	2.85	2.71	**381.30**	0.14	0.10
b4-24	514.81[1]	33.26	32.97	**506.74**	0.90	0.73
b4-32	**683.02**	54.16	52.83	**683.02**	4.46	3.80
b4-40	**843.31**	745.33	741.24	**843.31**	30.37	27.69
b4-48	-	T	-	**925.77**	108.03	97.54
Ave.		121.97			7.70	

[1] the algorithm stops as branching limit was reached

Table 3.3: Comparison of applying S1 and S2 on the Dantzig-Wolfe reformulation of the compact formulation presented in section 1.4.1 without the Consolidation Assumption

	S1			S2		
Instance	DB	T_m	T_{sp}	DB	T_m	T_{sp}
a2-16	**294.26**	62.14	61.67	**294.26**	95.15	94.99
a2-20	**344.86**	720.06	718.96	-	T	-
a2-24	**431.10**	2328.75	2326.26	-	T	-
a3-18	**300.49**	864.55	863.85	**300.49**	1630.52	1630.22
a3-24	**344.85**	3192.11	3190.25	-	T	-
a3-30	349.20[1]	3600.00	3600.00	-	T	-
a3-36	-	T	-	-	T	-
a4-16	**282.67**	2143.42	2143.03	**282.67**	109.05*	108.86
a4-24	354.56[1]	3600.00	3600.00	-	T	-
a4-32	-	T	-	-	T	-
a4-40	-	T	-	-	T	-
a4-48	-	T	-	-	T	-
b2-16	**309.39**	340.40	339.95	**309.39**	51.44	51.26
b2-20	**332.65**	170.83	169.61	**332.65**	8.21	7.88
b2-24	444.54[1]	1857.93	1855.71	444.54[1]	1389.32	1388.63
b3-18	301.41[1]	1179.01	1178.43	301.41[1]	15.48	15.27
b3-24	386.10[1]	3600.00	3600.00	-	T	-
b3-30	432.02[1]	3600.00	3600.00	-	T	-
b3-36	-	T	-	-	T	-
b4-16	**296.95**	118.83	118.43	**296.95**	1.30*	1.12
b4-24	284.66[1]	3600.00	3600.00	**371.39**	896.15	895.54
b4-32	360.78[1]	3600.00	3600.00	-	T	-
b4-40	-	T	-	-	T	-
b4-48	-	T	-	-	T	-
Ave.		1270.79			523.55	

[1] the algorithm stops as branching limit was reached

Table 3.4: Applying S3 and S4 on the Dantzig-Wolfe reformulation of the compact formulation presented in 1.4.1 without the Consolidation Assumption

	S3			S4		
Instance	DB	T_m	T_{sp}	DB	T_m	T_{sp}
a2-16	**294.26**	65.70	65.51	**294.26**	7.05	6.84
a2-20	322.54[1]	3600.00	3600.00	**344.86**	159.14	158.83
a2-24	-	T	-	**431.10**	2001.70	2000.87
a3-18	**300.49**	1348.23	1347.99	**300.49**	26.62	26.44
a3-24	-	T	-	**344.85**	3144.49	3143.90
a3-30	-	-	-	404.85[1]	3600.00	3600.00
a3-36	-	T	-	-	T	-
a4-16	**282.67**	87.79*	87.56	**282.67**	11.84*	11.70
a4-24	-	T	-	**375.02**	1374.98*	1374.54
a4-32	-	T	-	-	T	-
a4-40	-	T	-	-	T	-
a4-48	-	T	-	-	T	-
b2-16	**309.39**	56.96	56.78	**309.39**	23.01	22.82
b2-20	-	-	-	**332.65**	4.11	3.46
b2-24	444.54[1]	1271.92	1271.18	444.54[1]	51.39	50.49
b3-18	301.41[1]	14.45	14.21	301.41[1]	3.73	3.56
b3-24	-	T	-	-	T	-
b3-30	-	T	-	**531.43**	650.55	649.36
b3-36	-	T	-	-	T	-
b4-16	**296.95**	1.05*	0.90	**296.95**	0.47*	0.34
b4-24	**371.39**	447.93	447.48	**371.39**	87.29*	86.84
b4-32	-	T	-	**494.86**	375.11*	373.72
b4-40	-	T	-	-	T	-
b4-48	-	T	-	-	T	-
Ave.		411.75			26.43	

[1] the algorithm stops as branching limit was reached

Table 3.5: Applying S5 and S'4 to the Dantzig-Wolfe reformulation of the compact formulation presented in 1.4.1 without the Consolidation Assumption

	S5			S'4		
Instance	DB	T_m	T_{sp}	DB	T_m	T_{sp}
a2-16	**294.26**	93.27	92.31	**294.26**	7.05	6.84
a2-20	**344.86**	767.02	765.90	**344.86**	159.14	158.83
a2-24	**431.10**	1323.40	1321.62	**431.10**	2001.70	2000.87
a3-18	**300.49**	559.71	559.35	**300.49**	26.62	26.44
a3-24	**344.85**	3167.03	3165.63	**344.85**	3144.49	3143.90
a3-30	377.21[1]	3600.00	3600.00	439.20[1]	3600.00	3600.00
a3-36	-	T	-	-	T	-
a4-16	**282.67**	2041.75	2041.47	**282.67**	11.84*	11.70
a4-24	349.31[1]	3600.00	3600.00	**375.02**	1374.98*	1374.54
a4-32	-	T	-	-	T	-
a4-40	-	T	-	-	T	-
a4-48	-	T	-	-	T	-
b2-16	**309.39**	376.23	375.92	**309.39**	23.01	22.82
b2-20	**332.65**	102.64	101.71	**332.65**	4.11	3.46
b2-24	444.54[1]	1616.56	1614.75	**444.72**	66.44	65.39
b3-18	301.41[1]	1480.26	1479.89	**301.41**	6.24*	5.90
b3-24	382.54[1]	3600.00	3600.00	-	T	-
b3-30	403.42[1]	3600.00	3600.00	**531.43**	650.55	649.36
b3-36	461.09[1]	3600.00	3600.00	-	T	-
b4-16	**296.95**	70.31	70.03	**296.95**	0.47*	0.34
b4-24	313.79[1]	3600.00	3600.00	**371.39**	87.29*	86.84
b4-32	407.98[1]	3600.00	3600.00	**494.86**	375.11*	373.72
b4-40	334.94[1]	3600.00	3600.00	-	T	-
b4-48	-	T	-	-	T	-
Ave.		1229.76			28.62	

[1] the algorithm stops as branching limit was reached

4

Dynamic Dial-a-Ride Problem

Introduction

Over the past few years, the rapid growth in communication and information technologies, together with real-time fleet management, have attracted researchers to study vehicle routing problems in dynamic context.

In dynamic optimization, decisions must be taken in a changing environment (data are revealed over time). The ratio between the number of dynamic requests and the total number of requests together with how long in advance dynamic requests are known are two criteria that measure the degree of dynamism. Dynamic dial-a-ride problem is usually with low degree of dynamism as most requests are static and a few additional dynamic requests can sometimes be known somewhat in advance. As the routes must be adjusted in real time, developing solution approaches that integrate real-time information, while respecting the time limitations are challenging issue.

To study the dial-a-ride problem in a dynamic context, first we review some of the existing works dealing dynamic vehicle routing problems (VRPs) together with the solution approaches. Then, we discuss the dynamic dial-a-ride problem by outlining what dynamic aspects we have considered together with our solution approach whose correctness is tested numerically in section 4.3.

4.1. Literature Review

Dynamic VRPs are usually solved using a "**Rolling Time Horizon**" (RTH) framework [35]. Such methodology solves the static problem repeatedly over the updated data from the current time t to $t + L$, where L is the length of the horizon. Three different basic strategies can be distinguished under the framework of the RTH:

- The **first strategy** is to solve the static problem exactly each time new information is revealed. The algorithm for the static problem that is called on each re-optimization must accommodate initial conditions that reflect the impact of the past decisions. In the case that L is very large (long-term planning horizon) the re-optimization can be numerically demanding and hence a very efficient algorithm is needed for the static problem; If the re-optimization is too time consuming, the approach is inadequate in a real-time setting. On the other hand, a small L results in a myopic short-term schedule.

- The **second strategy** is to apply the static algorithm once only at the beginning of the planning horizon. On the arrival of new information, the current solution is updated using heuristic methods such as insertion heuristics, deletion heuristics, interchange moves, and fast local search procedure. The speed of heuristic approaches is useful not only to re-optimize quickly, but also to potentially decide about rejecting a new request: if it makes the problem infeasible or it yields a cost increase that is higher than the rejection cost.

- The **third strategy** is to apply hybrid schemes that take the advantage of the speed of the second strategy together with the solution quality of the first one. In particular, such an approach first applies a fast heuristic to impose updates. It then call a more sophisticated procedure to improve that solution.

Ichoua et al. [27] provided an excellent review on solution approaches for dynamic VRPs (including dynamic DARP) where the arrival of new requests is the only source of perturbations. Furthermore, Berbeglia et al. [8] provided a complete review of different contributions for dynamic pickup and delivery problems, where the case of dynamic DARP is also studied.

Another issue that must be taken into account for dynamic vehicle routing problems is how to determine the place of the vehicles at re-optimization time. There exists two alternatives to simulate the location of the vehicles along the route at re-optimization time:

1. To fix the place of the vehicle to its next immediate destination at re-optimization time; the majority of the existing contributions apply such idea.

2. Recent advances in communication and information technologies brought the idea of the second alternative, which is vehicle diversion (Regan et al. [37] and Ichoua et al. [25]).

Our approach to the dynamic dial-a-ride problem is to apply the first alternative: fixing the next destination of the vehicles at the time of re-optimization, under the framework of the first strategy: exact re-optimization.

There exists some techniques to better manage the dynamic aspect of the problem under consideration: forecasting the occurrence of new service requests through probability distributions obtained from historical data, and imposing their effect on solution methodologies. One way of applying such idea for VRPs with time windows is "waiting strategies". In particular, one needs to assess how much time a vehicle can wait at each node of its route before resuming the route (Bent and Hentenryck [7], Mitrović-Minić et al. [29], Branke et al. [10], Ichoua et al. [26], and Thomas [44]). "Buffering strategies" which hold a request for a while before assigning it to a vehicle are another approach (Pureza and Laporte [36]). Van Hemert and La Poutré [24] proposed the movement of vehicles to locations where the potential of occurrence of a new request is high. They called such a location a fruitful region. Bent and van Hentenryck [7] studied a partially dynamic VRPTW. They proposed a multiple scenario approach (MSA), which continuously generate and solve scenarios that include both static and dynamic requests. The goal is to leave room for accommodation of future requests in the plan. The real plans are then obtained by projection over actual requests only.

There exist some cases where no probability distribution is available to model future demands. Mitrović-Minić et al. [30] proposed double horizon approach, which extends the Rolling Time Horizon idea to incorporate both a

short-term and a long-term planning horizons. A different objective function is defined for each horizon. The objective function of the short-term horizon is the one associated with the static problem, while the objective of the long-term one favors large slack times in the routes to better accommodate future requests.

4.2. The Solution Approach

Before presenting our solution approach to solve the dynamic dial-a-ride problem, we categorize the possible perturbations into two groups:

- **Minor Perturbations**, which indicate the perturbations that happen at any time during the planning horizon; even if the serving of the request is started.
- **Major Perturbations**, which refer to the perturbations that happen before the beginning of the planning horizon. Indeed, sometimes the initial input data changes before executing any routes; but it is not possible to resolve the whole problem from scratch (for instance because the solver takes too much computing time). Hence, one would like to impose such changes dynamically.

The term perturbation refers to the arrival of new requests, requests cancelations, and requests modifications. Requests modifications refer to any changes in the time window, load and/or the place of the pickup (delivery) node of any request. In this study, we assume that requests cancelation and the arrival of totally new requests happen rarely. When we receive a totally new request as a minor perturbation, it is possible to either accept it or not. However, there are some cases where we must accept that request. Such situations can happen when the request has been declared well in advance, but it is mistakenly not considered to be covered in the static solution; therefore, they must be covered through the dynamic approach. There is not a lot of work in the literature that deals with cases where such different sources cause dynamism.

To re-optimize the problem after perturbation, we apply the branch-and-price approach presented in Chapter 3 for the static DARP under the Rolling Time Horizon framework. The on-line data are considered as an input to our

model as they are revealed, without considering any specific priority. The time at which the re-optimization is performed can be defined in two ways:

Review Policy A: executing re-optimization once new information is received for a request.

Review Policy B: executing re-optimization at a fixed review period where new information may have been received for a set of requests.

Applying any of the above alternatives may result in infeasibility, meaning that it is not possible to accept the new (set of) request(s). Assume that the second alternative is used and the problem is infeasible. If the set of updates contains one or several requests that must be covered, then such approach is not very efficient, in comparison to the first alternative. When the first alternative is applied and the problem is not feasible, then the planner can easily decide either to assign a taxi to the request, if it must be covered, or to reject the request.

The Dantzig-Wolfe reformulation that is applied in the process of the branch-and-price approach for dynamic DARP outlined below:

$$\min \quad \sum_{i=0}^{m-1} \sum_{g_i \in Z_i^{updated}} c_{g_i} \nu_{g_i} \tag{4.1}$$

$$\text{s.t}$$

$$\sum_{i=0}^{m-1} \sum_{g_i \in Z_i^{updated}} a_{jg_i} \nu_{g_i} = 1 \quad \forall j \in P'_{updated} \cup D'_{updated} \tag{4.2}$$

$$\sum_{g_i \in Z_i^{updated}} \nu_{g_i} = n_i \quad i = 0, ..., m-1, m, ..., \gamma - 1 \tag{4.3}$$

$$\nu_{g_i} \in \mathbb{Z}_+ \tag{4.4}$$

where c_{g_i} is the cost of the generated route g_i, and a_{jg_i} is one if route g_i visites node j; if not, $a_{jg_i} = 0$. $P'_{updated}$, $D'_{updated}$ and $Z_i^{updated}$ are the set of pickup nodes, delivery nodes and subproblem polyhedron that are defined over the updated data. These updated data can either include a new request or a new subset of requests. As it is depicted through constraint (4.3), for dynamic DARP we have γ new type of vehicles. At re-optimization time t,

some vehicles are servicing some requests. Each such vehicle must now be distinguished from other vehicle of the same type; hence it defines a new type of its own.

To determine the characteristics of these new type of vehicles at re-optimization time t, first we need to differentiate if a minor perturbation has happened or a major one. In the case that we have minor perturbation, we can use t as a feasible re-optimization time; otherwise, the re-optimization time is defined by the policy explained in next section, which is part of the random generation of input data.

Now we outline the procedure to identify the positioning of the vehicles at the feasible re-optimization time t, which is not trivial. Such procedure also includes the way we fix the already executed base-line solution at re-optimization time. For this purpose, each route of the base-line solution defined by the previous iterate optimized solution is memorized through a label whose definition is given in the process of the forward labeling approach in section 2.5. Let $\mathcal{L} = \{L_1, L_2, ..., L_n\}$ be the set of labels that record the vehicle routes of the current base-line solution. Assume that $\mathcal{P}(L)$ is the path corresponding to label L. Now one must identify the two consecutive positions j and l on $\mathcal{P}(L_i)$, for each $L_i \in \mathcal{L}$, for which we have $t \in (\mathrm{t}^{\mathcal{P}^j(L_i)}, \mathrm{t}^{\mathcal{P}^l(L_i)}]$, where $\mathrm{t}^{\mathcal{P}^j(L_i)}$ and $\mathrm{t}^{\mathcal{P}^l(L_i)}$ represents the start service time of the partial paths at position j and l of path $\mathcal{P}(L_i)$, respectively. If there is no perturbation at position l, the information at position l provide the characteristics of this new type of vehicle and it can be used to initialize the forward labeling approach described in section 2.5. In case there is some perturbation at l, an artificial index is used for the location of the vehicle, and $\mathrm{q}^{\mathcal{P}^j(L_i)}$, $\mathcal{V}^{\mathcal{P}^j(L_i)}$, and $\mathcal{O}^{\mathcal{P}^j(L_i)}$ provide the required information to initialize the dynamic programming approach.

Once the new type of vehicles are determined, one can apply the branch-and-price approach defined for the static problem on the updated data. The objective function could be modified to include some stability considerations. In the case that the re-optimization does not result in a feasible solution, some taxies will be assigned to the online requests that must be covered in the solution. The cost of using taxies will be added to the objective value of the base-line solution.

4.3. Numerical Experiments

4.3.1 Objectives of the Experiments

We are interested to numerically evaluate the performance of our dynamic optimization approach over the HellFest data. For this purpose the solution of the static, the dynamic, and the offline case are empirically compared. The offline solution refers to the one that is obtained by considering all the on-line information be available in advance. For this comparison the objective function (4.1) is used without any modification (no stability term is added). We wish to evaluate the optimality gap between the process of dynamic optimization and the idealistic case of the offline optimization.

4.3.2 Experimental Scheme

The scheme that is used for the experiments is to first apply the branch-and-price approach presented in Chapter 3 on the available data for the static problem at the beginning of the planning horizon. After that, at each re-optimization time, the branch-and-price approach is applied on an updated data set, while the new type of vehicles are determined. Note that in the process of the branch-and-price approach, VersionC of the forward labeling algorithm already discussed in Chapter 2 is used to solve the pricing problem. The updated data set is obtained via imposing the impact of a set of revealed information (up to re-optimization time) to the current input data (Review Policy B). After performing the last iteration of re-optimization, the actual data set of the problem is recorded that is used as an input for offline case. Finally, for the experiments of this chapter we have not considered the case where at least one on-line request must be covered in a mandatory way : all extra requests can be rejected (i.e. covered by a taxi).

4.3.3 Computational Results

A. Experimental Setting

The experiments of this chapter are also implemented in C++, under the environment of BaPCod and run on an MacBookPro5,2 (3,06 GHz) over a data set that includes 23 different instances. These instances are generated randomly in such a way that the real data for HellFest application is

simulated. In the rest of the text, we refer to this data set as dataSet2. The random generation procedure to generate the instances of the dataSet2 includes two phases. In the first phase, the data for the static case are generated. The possible perturbations are then simulated in the second phase.

For dataSet2, we know that the distance matrix (graph $\bar{G}$) is given, and $|\bar{N}| = 16$. We also know that there exist two types of vehicle at the beginning of the planning horizon: a vehicle with capacity 7 and 14 vehicles of capacity 9. To generate a random request r, a pickup node p_r and a delivery node d_r are randomly chosen from the set $\bar{N}$. Then, a random value $1 \leq q_r \leq Q$ is assigned to the load of the request. To generate the time windows for the request r, let $w = 10$ minutes, which is used to define the length of the time window. Then, a random value e_{p_r} is selected in such a way that $e_{p_r} \in [t_{o^+p_r}, T - t_{p_r d_r}]$, where $T = 1440$ minutes is the length of the planning horizon. Then, let $l_{p_r} = e_{p_r} + w$ and $\bar{l}_{p_r} = e_{p_r} + 2 * w$. For the delivery node of request r, define $e_{d_r} = e_{p_r} + t_{p_r d_r} + d_{p_r}$, $\bar{l}_{d_r} = \bar{l}_{p_r} + s_{p_r} + b_r^f$ and $l_{d_r} = (\bar{l}_{d_r} - e_{d_r})/2$. Note that the generated time window must be feasible, meaning that

$$\bar{l}_{d_r} + t_{d_r o^-} + s_{d_r} \leq \bar{l}_{o^-} \tag{4.5}$$

Finally, the service duration for the pickup node (and the delivery node) takes a random value in $[0, 15]$.

To simulate the possible perturbations over static instances generated in the first phase, we assume that at most five sets (blocks) of on-line information can be received. Each block of revealed information encompass both minor and major perturbations with the following frequency:

- At most five requests with "**time window**" modification.
- At most four requests with "**load**" modification.
- At most three requests with "**pickup/delivery location**" modification.
- At most two requests "**cancelation**".
- At most two new requests "**arrival**".

Note that any request can participate several times in either of above perturbations as long as it is not canceled. Besides, for some blocks, we can have at least two equal requests. Such a situation can be interpreted as load splitting.

Below the procedures to simulate the perturbations for a request, r, which is chosen randomly from an instance of the static dataSet2, are outlined:

Time Window Perturbation, the start service time, e_{p_r}, is increased by a random value between 10 minutes and 120 minutes. Besides, the length of the time windows are increased by a random value $\delta \in [0, w]$. Such time window perturbation is valid if the corresponding updated $\bar{l}_{p_r}$ is feasible.

Load Perturbation, the load of the random request, q_r, is increased by an integer random value in $[1, 3]$.

Pickup/Delivery Perturbation, a random vertex is selected from set V. Such vertex can be accepted provided that the corresponding time windows are feasible.

Using all the above procedures, a set of perturbed requests is generated for each instance of dataSet2 in such a way that it includes at most five blocks of perturbed requests. To each block a feasible re-optimization time is also assigned. The feasible re-optimization time for each block B is obtained as follows.

Let E be the set of the earliest start service time at pickup nodes of the requests belonging to block B:

$$E \;=\; \{e_{p_r} : r \text{ is a request in } B\}\ . \tag{4.6}$$

In the case where the perturbed request is with time window perturbation, the earliest start service time at the pickup node of the the static request is considered. The feasible re-optimization time t thus takes a random value in $[0, \min_E e_{p_r}]$.

B. Analysis of the Experiments

The numerical results are outlined in Table 4.1. As it can be seen the table is divided into three different blocks. The blocks represent the solution of the static case, the dynamic case, and the offline case. The first column of the table represent the id of the instances. In instance "a-b", "a" refers to the number of the requests that must be covered for the static case, and "b" indicates the seed that generated the random instance. The cost of the solution together with the required time to generate that solution are outlined in columns entitled by Cost and T, respectively. The reported times are in second. Note that the outlined results for dynamic case are associated to the last iteration of re-optimization.

As it was expected, the cost of solving the offline case is smaller or in the worse case equal to the dynamic case. For 34.78% of instances the cost of offline case is better than the dynamic counterpart, while the rest of the instances are solved with the same cost as the one for dynamic case. This can be understood because of the fact that at re-optimization time t, the vehicles that are servicing some requests cannot modify the currently passed path by t. Besides, they must complete servicing the requests that are open at time t. These are two restrictions in the process of dynamic optimization, that cause such difference in the value of the objective functions compared to the offline case. Furthermore, all new type of vehicles that have executed a route at re-optimization time must complete their route at least by returning back to depot.

Conclusion

In this chapter, the dial-a-ride problem is studied in a dynamic context. For this purpose, the branch-and-price approach already discussed in the third chapter is adjusted under the framework of a rolling time horizon approach. Different aspect of the scheme have been discussed and numerically experimented on a new data set that simulates the one of HellFest application. The numerical results have proved the correctness of our scheme. To the best of our knowledge this is the first work that applies the branch-and-price approach in a dynamic context for the dial-a-ride problem.

Table 4.1: Table of Results

	Static		Dynamic		Offline	
ID	Cost	T	Cost	T	Cost	T
22-23	17804	2.92	17751	3.74	17751	3.71
23-20	18806	3.76	20081	3.90	20081	3.11
26-14	20316	8.54	20842	9.35	20842	6.79
27-24	21285	5.87	21114	6.93	21114	7.50
27-29	21775	5.56	22352	7.24	22102	10.26
28-1	22442	7.88	22169	9.67	22169	5.86
28-28	21834	7.30	22167	8.60	22167	6.10
29-15	23734	8.90	24673	10.81	24423	8.27
29-17	21735	77.86	24055	9.39	24055	7.51
31-21	26000	9.11	25844	19.78	25844	13.48
31-27	28646	12.53	28833	9.89	28333	8.63
32-7	24291	13.68	24632	15.30	24382	14.15
32-25	25682	14.58	26318	25.97	26318	25.76
32-26	24668	8.83	25955	13.00	25777	9.50
33-2	24851	26.59	25586	24.29	25086	12.91
33-11	26341	12.32	24905	23.20	24655	24.51
35-0	28816	14.91	29156	25.78	29156	16.74
36-5	27517	25.65	27824	44.88	27824	34.83
36-16	26109	26.67	26661	19.60	26661	20.56
40-13	29370	30.13	29786	78.88	29286	35.92
40-18	29662	34.34	35113	93.76	35113	31.25
41-8	31275	35.30	31785	45.36	31785	44.75
41-9	30524	37.59	30827	25.85	30827	52.23

5

Dial-a-Ride Problem with Uncertainty

Introduction

First of all we would like to mention that a majority of the content of this section and section 5.1 is taken from the works of Thiele et al. [43] and Bertsimas and Sim [9].

Engineers, economists, investment professionals, and others need to make decisions to optimize a system, while information is incomplete and not reliable. Uncertainty is an inevitable feature of many decision-making environments. Conventional optimization techniques usually lead to solutions that can be easily disrupted, as the realized parameter values are usually different from the deterministic inputs. That is the reason why the solutions of deterministic models are rarely executed, and certainly, never truly optimal. Besides, there exists a high cost of recovery or repair. In this chapter, we are interested to study the dial-a-ride problem in the context of optimization under uncertainty to build solutions that are:

- less fragile to disruption.
- easier to repair if needed with minimum cost.

In general, there exist two alternatives to treat data uncertainty: stochastic programming and robust optimization.

A "stochastic programming" approach proceeds by constructing an explicit probability distribution for data and optimizing some stochastic variant of a priori objective function (expected value). From an informal robustness standpoint the reliance on an explicit probability distribution may be problematic.

"Robust optimization", in contrast, is a formal approach for optimization under uncertainty that relies on some uncertainty sets rather than a stochastic model of the distribution of data. Replacing probability distributions with uncertainty sets is equivalent to assigning a same weight to all possible data realizations.

Standard robust optimization (RO) formulations assume that the uncertain parameters will not be observed until all the decision variables are determined and therefore do not allow for recourse actions. Recourse actions may be based on realized values of some of the uncertain parameters. Multi-period decision models involve uncertain parameters some of which are revealed during the decision process. Therefore, a subset of the decision variables can be chosen after these parameters are observed in a way to correct the sub-optimality of the decisions made with less information in earlier stages. Adjustable robust optimization (ARO) formulations model such decision environments that allow recourse actions. These models are related to the two-stage (or multistage) stochastic programming formulations with recourse.

One contribution in the domain of robust optimization with recourse is the work of Thiele et al. [43]. They developed a robust approach for general two-stage linear optimization problems with uncertainty on the right-hand side only. They considered the following formulation as the robust counterpart of the original linear two-stage formulation:

$$\min \quad \mathbf{c^T x} \; + \; \max_{\mathbf{b}\in\mathcal{B}} \mathbf{Q}(\mathbf{x}, \mathbf{b}) \tag{5.1}$$

$$\text{s.t} \quad \mathbf{x} \; \in \; \mathbf{S} \tag{5.2}$$

where the uncertain parameters $\tilde{b}_i$ $(i = 1, ..., m)$ are defined using the poly-

hedral uncertainty sets described by Bertsimas and Sim [9]:

$$\mathcal{B} \;=\; \{\mathbf{b} : \tilde{b}_i = b_i + \hat{b}_i\, \delta_i, \sum_{i=1}^{m} |\delta_i| \leq \Gamma, |\delta_i| \leq 1\} \tag{5.3}$$

They assumed that the budget of uncertainty (the parameter Γ) is integer. If we assume that $\Gamma = 0$, the nominal problem is yielded, while $\Gamma = m$ corresponds to interval-based uncertainty sets and leads to the most conservative case.

As a feasible solution to a robust formulation must be feasible for any realization of the elements of the uncertainty set, robust optimization techniques consider the worst possible realization (worst-case scenario). Robust optimization is sometimes criticized for being overly conservative, and being robust only against data realizations that are allowed by the given uncertainty model, while potentially becoming very vulnerable to realizations outside of the realm of the model. Enlarging the uncertainty set is one alternative to treat the latter problem, but this of course may make the method even more conservative. Choosing uncertainty sets in such a way that they bring a good trade-off between performance and conservatism of the method is central to robust optimization approaches.

5.1. Single-Stage Robust Optimization

Traditionally, the term "robust optimization" refers to an approach for dealing with parameter uncertainty in single-stage optimization problems (problems without recourse). This approach was pioneered in 1973 by Soyster [42], who proposed a model that guarantees feasibility for all instances of the parameters within a convex set. To assure the tractability of the resulting model he considered a geometry for the uncertainty set that leads to very conservative solutions. The solutions are so conservative that they are too far from optimality of the nominal model. Hence, such approach is not of practical interest for real-life implementation.

The issue of over-conservatism of Soyster's approach hindered the adoption of robust techniques in optimization problems until the mid-1990s, when Ben-Tal and Nemirovski [4], [5], [6], El Ghaoui and Lebret [21] and El

Ghaoui et al. [22] started investigating tractable robust counterparts of linear, semidefinite and other important convex optimization problems. They focused on ellipsoidal uncertainty sets, which allow for important insights into the robust framework. However, the robust counterpart of linear models is second-order cone problems that increase the complexity of the problem considered. To manage the problem of complexity, Bertsimas and Sim [9] studied polyhedral uncertainty sets, which do not change the class of the problem at hand, and explicitly quantify the trade-off between performance and conservatism in terms of probabilistic bounds of constraint violation. Another advantage of their approach is that it can be easily extended to integer and mixed-integer programming problems.

Before reviewing the robust formulations proposed by Soyster (1973) [42], Ben-Tal and Nemirovski (2000) [6], and Bertsimas and Sim (2004) [9] for linear optimization, we emphasize on the fact that this part is directly extracted from the work of Bertsimas and Sim (2004) [9]. Now consider the following nominal linear program:

$$\max \quad \mathbf{c}'\mathbf{x} \tag{5.4}$$

$$\text{s.t}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{5.5}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{5.6}$$

In general, data uncertainty can arise on the elements of $\mathbf{c}'$, $\mathbf{A}$, and $\mathbf{b}$. First, we assume that data uncertainty affects only the elements of matrix $\mathbf{A}$. Note that the uncertainty of the objective function coefficients $\mathbf{c}'$ can be modeled as follows:

$$\max \quad z \tag{5.7}$$

$$\text{s.t}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{5.8}$$

$$z - \mathbf{c}'\mathbf{x} \leq \mathbf{0} \tag{5.9}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{5.10}$$

To model the data uncertainty, let J_i represent the set of coefficients in row i of matrix $\mathbf{A}$ that are subject to uncertainty. Each entry a_{ij}, $j \in J_i$ is

modeled as a symmetric and bounded random variable $\tilde{a}_{ij}$ that takes values in $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$. The random variable $\eta_{ij} = (\tilde{a}_{ij} - a_{ij})/\hat{a}_{ij}$ is associated with the uncertain data $\tilde{a}_{ij}$ such that it obeys an unknown but symmetric distribution, which takes values in $[-1, 1]$.

The Robust Formulation of Soyster [42] :

Under the model of data uncertainty described above, suppose that the uncertainty is column-wise, that is the column a_i of the constraint matrix in the constraints $\mathbf{Ax} \leq \mathbf{b}$, are known to belong to a given convex set, then the robust formulation is as follows:

$$\max \quad \mathbf{c}'\mathbf{x} \tag{5.11}$$

$$s.t$$

$$\sum_j a_{ij}\, x_j + \sum_{j \in J_i} \hat{a}_{ij}\, y_j \leq b_i \quad \forall i \tag{5.12}$$

$$-y_j \leq x_j \leq y_j \quad \forall j \tag{5.13}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{5.14}$$

$$\mathbf{y} \geq \mathbf{0} \tag{5.15}$$

Let $\mathbf{x}^*$ be the optimal solution of the above formulation. At optimality, $y_j = |x_j^*|$, and thus

$$\begin{aligned} \sum_j \tilde{a}_{ij}\, x_j^* &= \sum_j a_{ij}\, x_j^* + \sum_{j \in J_i} \eta_{ij}\, \hat{a}_{ij}\, x_j^* \\ &\leq \sum_j a_{ij}\, x_j^* + \sum_{j \in J_i} \hat{a}_{ij}\, |x_j^*| \leq b_i \quad \forall i \end{aligned} \tag{5.16}$$

which implies that for every possible realization $\tilde{a}_{ij}$ of the uncertain data, the solution remains feasible. In other words, the solution is *robust*. It is worth to note that for every ith constraint, the term, $\sum_{j \in J_i} \hat{a}_{ij}\, |x_j^*|$ gives the necessary *protection* of the constraint by maintaining a gap between $\sum_j a_{ij}\, x_j^*$ and b_i.

The Robust Formulation of Ben-Tal and Nemirovski [6] :

To address the conservatism of the former model, Ben-Tal and Nemirovski [6] proposed the following robust model:

$$\max \quad \mathbf{c}'\mathbf{x} \tag{5.17}$$

$$s.t$$

$$\sum_j a_{ij}\, x_j + \sum_{j\in J_i} \hat{a}_{ij}\, y_{ij} \tag{5.18}$$

$$+\Omega_i \sqrt{\sum_{j\in J_i} \hat{a}_{ij}^2\, z_{ij}^2} \;\leq\; b_i \quad \forall i \tag{5.19}$$

$$-y_{ij} \leq x_j - z_{ij} \;\leq\; y_{ij} \quad \forall i, j \in J_i \tag{5.20}$$

$$\mathbf{l} \leq \mathbf{x} \;\leq\; \mathbf{u} \tag{5.21}$$

$$\mathbf{y} \;\geq\; \mathbf{0} \tag{5.22}$$

Since every feasible solution of the former problem is a feasible solution to the latter problem, the robust model proposed by Ben-Tal and Nemirovski [6] is less conservative.

Assume that the original nominal problem has n variables and m constraints, while the bound constraints are not counted. Besides, let k coefficients of the $m \times n$ nominal matrix $\mathbf{A}$ be subject to uncertainty. The robust model proposed by Soyster [42] is a linear optimization problem with $2n$ variables, and $m + 2n$ constraints. In contrast, the model of Ben-Tal and Nemirovski [6] is a second-order cone problem, with $n + 2k$ variables and $m + 2k$ constraints. This latter model is non-linear; thus it is not attractive for solving robust discrete optimization models.

The Robust Formulation of Bertsimas and Sim [9] :

Bertsimas and Sim [9] proposed another approach that retains the advantages of the linear framework of Soyster [42] together with controlling the degree of conservatism for every constraint. Consider the ith constraint of the nominal problem $\mathbf{a_i'}\,\mathbf{x} \leq b_i$. Since the nature is restricted in its behavior, only a subset of the coefficients of this constraint may change in order to adversely affect the solution. It is unlikely that all of the a_{ij} $(j \in J_i)$ will change. Hence, for each constraint i, a parameter $\Gamma_i \in [0, |J_i|]$ is introduced,

which is not necessarily integer and it reflects the number of coefficients that are allowed to be uncertain. The goal is to be protected against all cases up to $\lfloor \Gamma_i \rfloor$ coefficients, and one coefficient a_{it} changes by $(\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it}$. In other words, the solution is guaranteed to be feasible if less than $\lfloor \Gamma_i \rfloor$ uncertain coefficients change. Moreover, they provide a probabilistic guarantee that even if more than $\lfloor \Gamma_i \rfloor$ coefficients change, then the robust solution will be feasible with high probability.

Now consider the following formulation, which is non-linear:

$$\max \quad \mathbf{c}'\mathbf{x} \tag{5.23}$$

$$s.t$$

$$\sum_j a_{ij}\, x_j \quad + \tag{5.24}$$

$$\max_{\{S_i \cup \{t_i\}: S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \{\sum_{j \in S_i} \hat{a}_{ij}\, y_j \quad + \tag{5.25}$$

$$(\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}\, y_t\} \quad \leq \quad b_i \quad \forall i \tag{5.26}$$

$$-y_j \leq x_j \quad \leq \quad y_j \quad \forall j \tag{5.27}$$

$$\mathbf{l} \leq \mathbf{x} \quad \leq \quad \mathbf{u} \tag{5.28}$$

$$\mathbf{y} \quad \geq \quad \mathbf{0} \tag{5.29}$$

Assuming that Γ_i is integer, the ith constraint will be protected by

$$\beta_i(\mathbf{x}, \Gamma_i) = \max_{\{S_i : S_i \subseteq J_i, |S_i| = \Gamma_i\}} \{\sum_{j \in S_i} \hat{a}_{ij}\, |x_j|\} \tag{5.30}$$

Note that if $\Gamma_i = 0$ then $\beta_i(\mathbf{x}, \Gamma_i) = 0$, which results in the nominal problem. Similarly, if $\Gamma_i = |J_i|$, then Soyster's approach is recovered. Therefore, by varying $\Gamma_i \in [0, |J_i|]$, we have the flexibility of adjusting the robustness of the method against the level of conservatism of the solution. Now the aim is to linearize the model. For this purpose, first we consider the protection function of the ith constraint for a given vector $\mathbf{x}^*$, which is as follows:

$$\beta_i(\mathbf{x}^*, \Gamma_i) = \max_{\{S_i \cup \{t_i\}: S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \{\sum_{j \in S_i} \hat{a}_{ij}\, |x_j^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor)\, \hat{a}_{it_i}\, |x_j^*|\} \tag{5.31}$$

Bertsimas and Sim [9] proved that (5.31) is equal to the objective function of the following linear optimization problem:

$$\begin{aligned} \beta_i(\mathbf{x}^*, \Gamma_i) &= \max \sum_{j \in J_i} \hat{a}_{ij} \, |x_j^*| \, \delta_{ij} && (5.32) \\ \text{s.t} & \\ \sum_{j \in J_i} \delta_{ij} &\leq \Gamma_i && (5.33) \\ 0 \leq \delta_{ij} &\leq 1 \quad \forall j \in J_i && (5.34) \end{aligned}$$

By substituting the dual formulation of model (5.32) to (5.34) into the non-linear model (5.23) to (5.29), the following linear program is obtained:

$$\begin{aligned} \max \quad & \mathbf{c}'\mathbf{x} && (5.35) \\ \text{s.t} & \\ \sum_j a_{ij} \, x_j + z_i \, \Gamma_i + \sum_{j \in J_i} p_{ij} &\leq b_i \quad \forall i && (5.36) \\ z_i + p_{ij} &\geq \hat{a}_{ij} \, y_j \quad \forall i, j \in J_i && (5.37) \\ -y_j \leq x_j &\leq y_j \quad \forall j && (5.38) \\ l_j \leq x_j &\leq u_j \quad \forall j && (5.39) \\ p_{ij} &\geq 0 \quad \forall i, j \in J_i && (5.40) \\ y_j &\geq 0 \quad \forall j && (5.41) \\ z_i &\geq 0 \quad \forall i && (5.42) \end{aligned}$$

The non-linear formulation (5.23) to (5.29) has $n + 2k$ variables and $m + 2k$ constraints. The robust counterpart model that is represented through conditions (5.35) to (5.42) has $n + k + 1$ variables and $m + k + n$ constraints, where $k = \sum_i |J_i|$ is the number of uncertain parameters.

In most real-world applications, the matrix $\mathbf{A}$ is sparse. Considering the above results, the linear formulation (5.35) to (5.42) preserves the sparsity of matrix $\mathbf{A}$. Finally, the obtained robust linear formulation (5.35) to (5.42) can be solved in the same way that the nominal problem can be solved. The solutions to formulation (5.35) to (5.42) are robust, as they are more likely to be executed or easier to be repaired when disruption happens. In the following the approach to model the right-hand side uncertainty is outlined.

5.2. Application to the Dial-a-Ride Problem with Uncertain Time Windows

Now we are interested to study the Dial-a-Ride Problem with Uncertain Time Windows (DARPUTW). When the given time windows are not reliable, the planner may be tempted to under-estimate the length of the time windows. Our goal is to apply a robust optimization approach that allows to produce more reliable solutions. Note that the discussion herein does not consider the decision variables and constraints associated to tardiness of the vehicle. Before discussing our method, we review the relevant literature for the vehicle routing application.

In 2013, Pillac et al. [33] provided a complete review on dynamic vehicle routing problems. As they mentioned, all type of uncertainties that are addressed in this application refer to demand uncertainty and distance travel uncertainty. Among these contributions, the majority of them apply stochastic programming approaches. Agra et al. [1] was the first who proposed a general approach to the robust vehicle routing problem with time windows and uncertain travel time. They worked with the classical framework of the single-stage robust programming: the uncertain travel times is modeled using the polyhedral uncertainty sets. Heilporn et al. [23] studied the single vehicle DARP with stochastic customer delays. They used an integer L-shaped algorithm, which is a stochastic programming technique to solve this problem.

We are interested to apply the classical framework of the single-stage robust programming to model the uncertainty for the "earliest start service times of pickup nodes" in the dial-a-ride application. In this case not only the uncertainty must be taken into account, but also the routing problem must be solved. To the best of our knowledge there is no previous work that models the possible delays using polyhedral uncertainty sets.

As it was explained, Bertsimas and Sim [9] assumed that for each constraint i a subset of coefficients a_{ij} is uncertain. However, in our case each relevant constraint has only two coefficients, one of which is uncertain, while such situation happens to a subset of constraints.

To model data uncertainty for time windows, first we need to classify the possible perturbations that can happen to the earliest start service times. In fact, there exist two main categories:

Minor Perturbation which is a bounded perturbation. Such perturbation can be interpreted as a reasonable amount of change in the lower bound of the time windows.

Major Perturbation which is an unbounded perturbation. Such perturbation can be interpreted as a cancelation or arrival of a totally new demand.

To define the uncertainty set, we assume that all uncertainty is caused by minor perturbations, as the major ones can be considered in the context of dynamic optimization. Let $[e_{p_r}, \bar{l}_{p_r}]$ be the original time window associated to the pickup node of request $r \in \mathcal{R}$. The uncertainty set can be defined in two ways:

1. $\tilde{e}_{p_r} \in [e_{p_r}, e_{p_r} + \hat{e}_{p_r}]$ if $e_{p_r} + \hat{e}_{p_r} < \bar{l}_{o^-}$;
2. $\tilde{e}_{p_r} \in [e_{p_r} - \hat{e}_{p_r}, e_{p_r} + \hat{e}_{p_r}]$ if $e_{p_r} + \hat{e}_{p_r} < \bar{l}_{o^-}$ and $e_{p_r} - \hat{e}_{p_r} > e_{o^+}$;

We assume that only delays can happen to the earliest start service time of pickup nodes, which is modeled through the first case. The budget uncertainty polytope is then as follows:

$$\mathcal{B} = \{(\tilde{e}_{p_r})_{r\in\mathcal{R}} : \tilde{e}_{p_r} = e_{p_r} + \hat{e}_{p_r}\ \delta_{p_r}, \sum_{r\in\mathcal{R}} \delta_{p_r} \leq \Gamma, 0 \leq \delta_{p_r} \leq 1, r \in \mathcal{R}\} \quad (5.43)$$

Note that for each request $r \in \mathcal{R}$, the earliest start service time at the delivery node of the request, e_{d_r}, and the deadline at pickup node and delivery node of the request, $(\bar{l}_{p_r}, \bar{l}_{d_r})$, can be defined using e_{p_r} as follows:

$$\bar{l}_{p_r} = e_{p_r} + \Delta_0^r$$

$$e_{d_r} = e_{p_r} + \Delta_1^r$$

$$\bar{l}_{d_r} = e_{p_r} + \Delta_2^r$$

Hence, for each request $r \in \mathcal{R}$ and vehicle $k \in K$ and $\tilde{e} \in \mathcal{B}$, the deterministic time window constraints are modified as follows:

$$\begin{aligned} \tilde{e}_{p_r}\, y_r^k &\leq t_{p_r}^k \leq (\tilde{e}_{p_r} + \Delta_0^r)\, y_r^k \\ (\tilde{e}_{p_r} + \Delta_1^r)\, y_r^k &\leq t_{d_r}^k \leq (\tilde{e}_{p_r} + \Delta_2^r)\, y_r^k \end{aligned}$$

Suppose that set Λ includes the extreme points of the following two inequalities:

$$\begin{aligned} \sum_{r \in \mathcal{R}} \delta_{p_r} &\leq \Gamma \\ 0 \leq \delta_{p_r} &\leq 1 \qquad \forall r \in \mathcal{R} \end{aligned}$$

Then for each $r \in \mathcal{R}, k \in K$ and $\tilde{\delta}^\omega \in \Lambda$ we have:

$$\begin{aligned} e_{p_r}\, y_r^k + (\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega &\leq t_{p_r}^k \leq (e_{p_r} + \Delta_0^r)\, y_r^k + (\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega \\ (e_{p_r} + \Delta_1^r)\, y_r^k + (\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega &\leq t_{d_r}^k \leq (e_{p_r} + \Delta_2^r)\, y_r^k + (\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega \end{aligned}$$

which are equivalent to

$$\begin{aligned} (\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega &\leq \min\{t_{p_r}^k - e_{p_r}\, y_r^k, t_{d_r}^k - (e_{p_r} + \Delta_1^r)\, y_r^k\} \\ (\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega &\geq \max\{t_{p_r}^k - (e_{p_r} + \Delta_0^r)\, y_r^k, t_{d_r}^k - (e_{p_r} + \Delta_2^r)\, y_r^k\} \end{aligned}$$

The robust model is obtained by replacing the time window constraints of static model with above obtained relations.

$$\min \quad \sum_{k\in K}\{\sum_{i\in N}\sum_{j\in N} c_{ij}^k x_{ij}^k + \beta \sum_{r\in\mathcal{R}} c_r^f f_r^k\} \tag{5.44}$$

$$\sum_{k\in K} y_r^k = 1 \quad \forall r \in \mathcal{R} \tag{5.45}$$

$\forall k \in K$:

$$y_r^k - \sum_{j\in N\setminus\{o^+\}} x_{p_r j}^k = 0 \quad \forall r \in \mathcal{R} \tag{5.46}$$

$$y_r^k - \sum_{i\in N\setminus\{o^-\}} x_{i d_r}^k = 0 \quad \forall r \in \mathcal{R} \tag{5.47}$$

$$\sum_{j\in P'} x_{o^+ j}^k = 1 \tag{5.48}$$

$$\sum_{j\in N\setminus\{o^-\}} x_{ji}^k - \sum_{j\in N\setminus\{o^+\}} x_{ij}^k = 0 \quad \forall i \in N\setminus O' \tag{5.49}$$

$$\sum_{i\in D'} x_{i o^-}^k = 1 \tag{5.50}$$

$$t_i^k + s_i + t_{ij} - M_{ij}^k(1-x_{ij}^k) \le t_j^k \quad \forall i \in N\setminus\{o^-\}, j \in N\setminus\{o^+\} \tag{5.51}$$

$$Q_i^k + q_j - W_{ij}^k(1-x_{ij}^k) \le Q_j^k \quad \forall i \in N\setminus\{o^-\}, j \in N\setminus\{o^+\} \tag{5.52}$$

$$q_r\, y_r^k \le Q_{p_r}^k \le Q_k\, y_r^k \quad \forall r \in \mathcal{R} \tag{5.53}$$

$$0 \le Q_{d_r}^k \le (Q_k - q_{d_r})\, y_r^k \quad \forall r \in \mathcal{R} \tag{5.54}$$

$$t_{d_r}^k - t_{p_r}^k - s_{p_r} \le f_r^k \quad \forall r \in \mathcal{R} \tag{5.55}$$

$$t_{p_r d_r}\, y_r^k \le f_r^k \le b_r^f\, y_r^k \quad \forall r \in \mathcal{R} \tag{5.56}$$

$\forall r \in \mathcal{R}, \tilde{\delta}^\omega \in \Lambda$:

$$(\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega \le \min\{t_{p_r}^k - e_{p_r}\, y_r^k, t_{d_r}^k - (e_{p_r} + \Delta_1^r)\, y_r^k\} \tag{5.57}$$

$$(\hat{e}_{p_r}\, y_r^k)\, \tilde{\delta}_{p_r}^\omega \ge \max\{t_{p_r}^k - (e_{p_r} + \Delta_0^r)\, y_r^k, t_{d_r}^k - (e_{p_r} + \Delta_2^r)\, y_r^k\} \tag{5.58}$$

Our current branch-and-price scheme that is developed in Chapter 3 cannot be directly used to solve the obtained robust model, as handling constraints (5.57) and (5.58) at once is not possible. To solve this problem one needs to develop a column-and-row generation approach.

Conclusion

In this chapter, the single-stage robust optimization framework has been used to study the dial-a-ride problem with uncertainty. We have assumed that the uncertainty happens to the earliest start service time. To the best of our knowledge this is the first model where such parameter uncertainty is considered. Besides, the structure of the uncertainty led us to define the polyhedral uncertainty set based on the combination of the one proposed by Bertsimas and Sim [9] and Thiele et al. [43]. Finally we have highlighted that the column generation approach used for the static and dynamic models does not extend to the robust optimization model.

Conclusion

In this thesis the heterogenous dial-a-ride problem has been studied in the static, the dynamic, and the robust context. The relevant mathematical formulations have been presented in Chapter 1 for the static case under the request consolidation assumption (that reduces the problem to the vehicle routing problem with time windows) and without it. Solving the static dial-a-ride problem without any specific assumption using the column generation approach raised very interesting challenges because of the existence of the ride time constraints and costs.

In Chapter 2, the issue of the ride time constraints in the process of the dynamic programming solution of the pricing subproblem has been discussed in detail, and a procedure to manage them has been proposed. Managing the ride time constraints in the process of the dynamic programming is one of the contributions of this thesis. Besides, the performance of different variants of the forward labeling algorithm has been theoretically and numerically assessed. Our numerical results show that applying state-space relaxation together with partial dominance rule works better than other techniques. It has allowed us to improve over existing benchmarks for the pickup and delivery problem with time windows proposed by Robke and Cordeau [38].

In Chapter 3, the branch-and-price approach has been applied to the Dantzig-Wolfe reformulation of the compact formulations of Chapter 1. Under the request consolidation assumption, all the existing challenging instances in the literature have been solved to optimality in a very short amount of time, but two of these for which the assumption has resulted in infeasibility, because the number of available vehicles are limited. For the unrestricted case, there exists no previously published work on column

generation approach to which we can compare our results. Indeed, handling the ride time constraints in the process of the forward labeling approach to the pricing problem is quite difficult. Our work brings some contributions in this direction. Moreover, different accelerating techniques have also been discussed and experimentally tested. Hence, we compare our results with the best solutions in the existing literature on branch-and-cut approaches.

According to the numerical results of Chapter 3, combining versionD of forward labeling algorithm with strong degree cuts, outperforms the other schemes that have been theoretically discussed and numerically tested in this chapter. While this scheme has allowed us to solve six instances much faster than the branch-and-cut approach of Cordeau [12], on average it is time consuming due to our procedure to recompute ride times along the partial paths in the process of dynamic programming. In particular, 62.5% of instances have been solved to optimality in our one hour time limit, while the branch-and-cut approach of Cordeau [12] has solved 79.17% of the instances to optimality, in a four hour time limit.

The branch-and-price approach has then been adjusted for the dynamic dial-a-ride problem. Applying the branch-and-price approach for re-optimization under the framework of the rolling time horizon approach has brought some issues that have been properly addressed and numerically tested in Chapter 4. These issues are:

- How to define a feasible re-optimization time.
- Strategies to handle the on-line information: one at a time on arrival, or waiting to receive several updates.
- How to handle on-going routes by defining new types of vehicles and managing them in the process of dynamic programming.

To the best of our knowledge this is the first work that applies branch-and-price approach in dynamic context for the dial-a-ride problem.

Finally, Chapter 5 brings some contributions on the ways the uncertainty can be modeled in robust approaches. We have assumed that the uncertainty carries only on the time windows. Then, we outline alternative models for the uncertainty set and the resulting static robust models without recourse

actions. We have found no previous work on this topic. Further work in this line of research would be to use such robust approach to build a base line solution in a rolling time horizon context in comparison to the static deterministic model that we have used in Chapter 4. The robust model that we obtained is however not simply amenable to the column generation approach developed in Chapter 3. Hence, another issue for further research is to develop a column-and-row generation approach that can solve the obtained robust model.

Bibliography

[1] A. Agra, M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss, and C. Requejo, *Layered formulation for the robust vehicle routing problem with time windows*, in Combinatorial Optimization, Springer, 2012, pp. 249–260.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[3] R. Baldacci, A. Mingozzi, and R. Roberti, *New route relaxation and pricing strategies for the vehicle routing problem*, Operations research, 59 (2011), pp. 1269–1283.

[4] A. Ben-Tal and A. Nemirovski, *Robust convex optimization*, Mathematics of Operations Research, 23 (1998), pp. pp. 769–805.

[5] A. Ben-Tal and A. Nemirovski, *Robust solutions of uncertain linear programs*, Operations Research Letters, 25 (1999), pp. 1 – 13.

[6] A. Ben-Tal and A. Nemirovski, *Robust solutions of linear programming problems contaminated with uncertain data*, Mathematical Programming, 88 (2000), pp. 411–424.

[7] R. W. Bent and P. V. Hentenryck, *Scenario-based planning for partially dynamic vehicle routing with stochastic customers*, Operations Research, 52 (2004), pp. 977–987.

[8] G. Berbeglia, J. . Cordeau, and G. Laporte, *Dynamic pickup and delivery problems*, European Journal of Operational Research, 202 (2010), pp. 8–15. Cited By (since 1996): 40.

[9] D. Bertsimas and M. Sim, *The price of robustness*, Operations research, 52 (2004), pp. 35–53.

[10] J. Branke, M. Middendorf, G. Noeth, and M. Dessouky, *Waiting strategies for dynamic vehicle routing*, Transportation science, 39 (2005), pp. 298–312.

[11] C. Contardo, B. Gendron, and J.-F. Cordeau, *A branch-and-cut-and-price algorithm for the capacitated location-routing problem*, CIRRELT, 2011.

[12] J.-F. Cordeau, *A branch-and-cut algorithm for the dial-a-ride problem*, Operations Research, 54 (2006), pp. 573–586.

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

[14] M. Desrochers and F. Soumis, *A generalized permanent labelling algorithm for the shortest path problem with time windows*, Inform, 26 (1988), pp. 191–212.

[15] J. Desrosiers, P. Pelletier, and F. Soumis, *Plus court chemin avec contraintes d"horaires*, (1983).

[16] J. Desrosiers, F. Soumis, and M. Desrochers, *Routing with time windows by column generation*, Networks, 14 (1984), pp. 545–565.

[17] B. Detienne, N. Rahmani, R. Sadykov, and F. Vanderbeck, *Optimizing timing decisions in dial-a-ride problems*, working paper (in preparation), Institute of Mathematics, University of Bordeaux, (July 2014).

[18] D. Feillet, *A tutorial on column generation and branch-and-price for vehicle routing problems*, 4OR, 8 (2010), pp. 407–424.

[19] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, *An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems*, Networks, 44 (2004), pp. 216–229.

[20] S. Gélinas, M. Desrochers, J. Desrosiers, and M. Solomon, *A new branching strategy for time constrained routing problems with application to backhauling*, Annals of Operations Research, 61 (1995), pp. 91–109.

[21] L. E. Ghaoui and H. Lebret, *Robust solutions to least-squares problems with uncertain data*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1035–1064.

[22] L. E. Ghaoui, F. Oustry, and H. Lebret, *Robust solutions to uncertain semidefinite programs*, SIAM J. OPTIMIZATION, 9 (1998), pp. 33–52.

[23] G. Heilporn, J.-F. Cordeau, and G. Laporte, *An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays*, Discrete Applied Mathematics, 159 (2011), pp. 883–895.

[24] J. Hemert and J. Poutré, *Dynamic routing problems with fruitful regions: Models and evolutionary computation*, in Parallel Problem Solving from Nature - PPSN VIII, X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervós, J. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, eds., vol. 3242 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 692–701.

[25] S. Ichoua, M. Gendreau, and J.-Y. Potvin, *Diversion issues in real-time vehicle dispatching*, Transportation Science, 34 (2000), pp. 426–438.

[26] ——, *Exploiting knowledge about future demands for real-time vehicle dispatching*, Transportation Science, 40 (2006), pp. 211–225.

[27] ——, *Planned route optimization for real-time vehicle routing*, in Dynamic Fleet Management, V. Zeimpekis, C. Tarantilis, G. Giaglis, and I. Minis, eds., vol. 38 of Operations Research/Computer Science Interfaces Series, Springer US, 2007, pp. 1–18.

[28] M. Junger and C. O. Workshop, *50 years of integer programming, 1958-2008 from the early years to the state-of-the-art*, Springer, Heidelberg; New York, 2010.

[29] Mitrović-Minić and G. Laporte, *Waiting strategies for the dynamic pickup and delivery problem with time windows*, Transportation Research Part B: Methodological, 38 (2004), pp. 635 – 655.

[30] S. Mitrović-Minić, R. Krishnamurti, and G. Laporte, *Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows*, Transportation Research Part B: Methodological, 38 (2004), pp. 669 – 685.

[31] S. Parragh, J.-F. Cordeau, K. Doerner, and R. Hartl, *Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints*, OR Spectrum, 34 (2012), pp. 593–633.

[32] S. N. Parragh, *Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem*, Transportation Research Part C: Emerging Technologies, 19 (2011), pp. 912–930.

[33] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, *A review of dynamic vehicle routing problems*, European Journal of Operational Research, 225 (2013), pp. 1 – 11.

[34] W. B. Powell and Z.-L. Chen, *A generalized threshold algorithm for the shortest path problem with time windows*, (1998).

[35] H. N. Psaraftis, *Dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem*, Mathematics of Operations Research, 14 (1980), pp. 130–154.

[36] V. Pureza and G. Laporte, *Waiting and buffering strategies for the dynamic pickup and delivery problems with time windows*, INFOR, 46 (2008), pp. 165–175.

[37] A. C. Regan, H. S. Mahmassani, and P. Jaillet, *Improving efficiency of commercial vehicle operations using real-time information: potential uses and assignment strategies*, Transportation Research Record, (1995), pp. 188–198. cited By (since 1996)38.

[38] S. Ropke and J.-F. Cordeau, *Branch and cut and price for the pickup and delivery problem with time windows*, Transportation Science, 43 (2009), pp. 267–286.

[39] S. Ropke, J.-F. Cordeau, and G. Laporte, *Models and branch-and-cut algorithms for pickup and delivery problems with time windows*, Networks, 49 (2007), pp. 258–272.

[40] M. Sigurd, D. Pisinger, and M. Sig, *Scheduling transportation of live animals to avoid the spread of diseases*, Transportation Science, 38 (2004), pp. 197–209.

[41] M. Sol, *Column Generation Techniques for Pickup and Delivery Problems*, PhD thesis, Technische Universitet Eindhoven, 1994.

[42] A. L. Soyster, *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Operations Research, 21 (1973), pp. 1154–1157.

[43] A. Thiele, T. Terry, and M. Epelman, *Robust linear optimization with recourse*, Rapport technique, (2009), pp. 4–37.

[44] B. W. Thomas, *Waiting strategies for anticipating service requests from known customer locations*, Transportation Science, 41 (2007), pp. 319–331.

[45] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*, SIAM, Philadephia, To be appear.

[46] F. Vanderbeck, *BaPCod: a generic branch-and-price code*, depot APP 08-120018-000 IDDN, http://wiki.bordeaux.inria.fr/realopt/, 2008.

[47] F. Vanderbeck, *Branching in branch-and-price: a generic scheme*, Mathematical Programming, 130 (2011), pp. 249–294.