UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO-PPGEP

Optimization models and solution methods for the Vehicle Allocation Problem

Cesar Dario Alvarez Cruz

São Carlos
December 2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO-PPGEP

Optimization models and solution methods for the Vehicle Allocation Problem

Cesar Dario Alvarez Cruz

Advisor: Prof. Dr. Reinaldo Morabito Neto
Co-Advisor: Prof. Dr. Pedro Augusto Munari Junior

Texto para defesa de tese apresentado ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de São Carlos como requisito para a obtenção do título de Doutor em Engenharia de Produção.

São Carlos
Monday 20th December, 2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

Folha de Aprovação

Defesa de Tese de Doutorado do candidato Cesar Dario Alvarez Cruz, realizada em 21/10/2021.

Comissão Julgadora:

Prof. Dr. Reinaldo Morabito Neto (UFSCar)

Prof. Dr. Pedro Augusto Munari Junior (UFSCar)

Prof. Dr. Victor Claudio Bento de Camargo (UFSCar)

Profa. Dra. Franklina Maria Bragion de Toledo (USP)

Prof. Dr. Alysson Machado Costa (University of Melbourne)

Prof. Dr. Denis Borenstein (UFRGS)

Prof. Dr. Leandro Callegari Coelho (ULaval)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.
O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Engenharia de Produção.

To my parents, Esperanza and Juan,

to my beloved siblings, Laura and Juan Manuel.,

to my nephew, Juan Jose,

to my beautiful girlfriend Vivi

You are the ones who gave me the strength to keep going through this process and to you I dedicate this work.

Acknowledgments

To that immensity people call God, the one that rules fate or let fate rules in its own way, I am deeply thankful for fating me this beautiful journey along the most amazing people I have ever met.

To my parents, Esperanza and Juan Paulo, my beloved siblings, Juan Manuel and Laura, and my nephew, Jojo, I am thankful for giving me your unconditional support and love. Without it, it would have been impossible to find meaning in all the things I do, among them, this work.

To my super cool and more than amazing supervisors, Reinaldo and Pedro, and unofficial supervisor, Alysson, I am thankful for your support and friendship. It is clear I admire you all because of how competent and intelligent you are. Yet, I admire you even more because of your integrity and the high standards for ethics by which you abide in your professional life. I can only hope to encounter people at least as amazing as you in my future professional life.

To my beloved partner, Vivi, your love and company was crucial for feeling happiness in the last four years.

To my Aunt Myriam and the Loaiza family, I am thankful for keeping me in your prayers.

Finally, I thank the Brazilian Agencies, CNPq and CAPES, that supported me financially during this time.

"Life can only be understood backwards, but it must be lived forwards."

Soren Kierkegaard

Resumo

O Problema de Alocação de Veículos (VAP) consiste em alocar uma frota de veículos para atender a demanda por serviços de transporte de carga entre terminais ao longo de um horizonte de planejamento. O objetivo é maximizar os lucros gerados pelos serviços completados. Prévias abordagens determinísticas e estocásticas utilizaram procedimentos heurísticos e de aproximação para resolver instâncias de grande porte para o problema. Esta tese contribui com modelos e métodos de solução exatos para resolver efetivamente instâncias do VAP de grande porte.

O primeiro método é um algoritmo *Branch-and-Benders-Cut* para resolver a formulação baseada na rede de espaço-tempo do VAP. A reformulação de Benders resulta num subproblema com estrutura de Problema de Fluxo de Custo Mínimo para cada tipo de veículo onde o fluxo é constituido por veículos vazios exclusivamente. Nós propomos duas desigualdades válidas para tentar reduzir o número de cortes de factibilidade e otimalidade necessários para atingir a solução ótima. Adicionalmente, utilizamos algoritmos de fluxo em redes para acelerar o processo de geração de cortes. Experimentos computacionais são mostrados para instâncias geradas aleatóriamente.

O segundo método é um algoritmo exato do tipo *Branch-and-Price* (BP), o qual proporciona soluções ótimas ou certificados de qualidade para resolver problemas de grande porte em tempos computacionais razoáveis. Este método é o resultado de reformular o modelo compacto de Programação Linear Inteira do VAP por meio da reformulação *Dantzig-Wolfe* e utilizar procedimentos eficientes para tratar cada componente da reformulação. O Método de Geração de Colunas Primal-Dual (PDCGM) é usado para resolver o problema mestre, enquanto o subproblema é modelado como um Problema de Fluxo de Custo Máximo e resolvido via agregação de soluções ótimas de caminhos máximos em Grafos Acíclicos Direcionados (DAG). Finalmente, propomos três procedimentos de ramificação para obter a solução ótima inteira do VAP. Experimentos computacionais com instâncias de um estudo de caso e instâncias aleatórias de tamanho realista são apresentadas e analisadas, o qual mostra a superioridade do método proposto quando comparado com outros métodos exatos para resolver instâncias de grande porte do VAP.

O terceiro método está baseado em preprocessar o grafo de espaço-tempo e reformular o problema em termos de quantos veículos vazios rotear entre os nós de demanda (arcos no modelo prévio). O

tamanho do modelo resultante depende do número de nós de demanda e o tamanho da frota, o qual pode ser vantajoso quando o número de pares terminal-períodos na rede de espaço-tempo é grande comparado com o número de arcos de demanda. Nos propomos um método BP baseado na reformulação *Dantzig-Wolfe* deste novo modelo. Os resultados de ambas, a reformulação resolvida com um solver de proposito geral e o BP, mostram a superioridade desta nova abordagem para resolver instâncias de tamanho realista para o VAP.

Palavras-chave: Problema de Alocação de Veículos, decomposição de Benders, decomposição de Dantzig-Wolfe, geração de colunas, transporte rodoviário de carga, logística.

Abstract

The Vehicle Allocation Problem (VAP) consists in allocating a fleet of vehicles to attend the expected demand for road freight transportation between terminals along a finite multiperiod planning horizon. The objective is to maximize the profits generated for the completed services. Previous deterministic and stochastic approaches used heuristic procedures and approximation methods for solving large scale instances of this problem. This thesis contributes with models and solution methods for solving effectively large-scale instances of the VAP.

The first method is Branch-and-Benders-Cut (BBC) for solving the space-time network formulation of the VAP. The Benders reformulation results in each subproblem being a multiple origin-destination minimum cost flow problem among empty vehicles exclusively. We propose two valid inequalities in order to reduce the number of infeasible cuts needed to reach a feasible and optimal solution. In addition, we use network flow algorithms in trying to accelerate the process of cut generation. Computational results are shown for randomly generated instances.

The second method is a tailored exact Branch-and-Price (BP) procedure, that provides optimal solutions or certificates of quality, for solving large-scale problems within reasonable computational times. This method is the result of reformulating a compact Integer Linear Programming model of the VAP through the Dantzig-Wolfe (DW) decomposition and using efficient procedures for solving each component of the reformulation. The Primal Dual Column Generation Method (PDCGM) is used to solve the master problem, while the subproblem is modeled as a Maximum Cost Flow Problem and solved using the aggregation of optimal longest paths problems on Directed Acyclic Graphs (DAG). Finally, we resort to three branching procedures to obtain the optimal integer solution of the VAP. Computational experiments with instances from a case study and random realistic-sized instances are presented and analyzed, showing that the method has a superior performance with respect to other exact approaches in solving large-scale VAP instances.

The third method is based on preprocessing the time-space extended graph and reformulating the problem in terms of routing empty vehicles along demand nodes. The resulting model's size depends on the number of demand nodes (arcs in the previous model) and fleet size, which can be advantageous when the number of terminal-period pairs in the time-space extended network is large compared to

the actual number of loads requested. We propose a BP method based on the DW reformulation of this new modelling approach. The results of both, the reformulation solved by CPLEX and the BP, shows the superior performance of this new approach in solving realistic-sized instances of the VAP.
Key words: Vehicle Allocation Problem, Dantzig-Wolfe decomposition, Benders decomposition, column generation, road freight transportation, logistics.

List of Figures

List of Tables

Contents

Acronyms

ACCPM Analytical Center Cutting Plane Method

ACO Ant Colony Optimization

BB Branch-and-Bound

BBC Branch-and-Benders-Cut

BC Branch-and-Cut

BP Branch-and-Price

BPC Branch-and-Price-and-Cut

CG Column Generation

CNDP Capacitated Network Design Problem

DAG Directed Acyclic Graph

DSS Decision Support System

DW Dantzig-Wolfe

EDI Electronic Data Interchange

GCSPPLG Generalized Cardinality-Constrained Shortest Path Problem on a layered graph

GPS Global Positioning System

GRASP Greedy Randomized Adaptive Search Procedure

ILP Integer Linear Programming

LP Linear Programming

LQN Logistics Queueing Network

LTL Less-than-Truckload

MCFP Multicommodity Network Flow Problem

MCFP Minimum Cost Flow Problem

MFP Maximum Flow Problem

MIP Mixed Integer Programming

MP Master Problem

NSA Network Simplex Algorithm

PDCGM Primal Dual Column Generation

RMP Restricted Master Problem

SA Simulated Annealing

SCAM Successive Convex Approximation Method

SLAM Successive Linear Approximation Method

SREDM Successive Resource-Directive Decomposition Method

TL Truckload

TSP Travelling Salesman Problem

USA United States of America

VAP Vehicle Allocation Problem

Chapter 1

Introduction

Freight transportation plays an important role in supply chains and the overall economy, since it supports production, trade and consumption activities by ensuring the effective movement of available resources, ranging from raw materials to finished goods. Thus, it is not a surprise that transportation accounts for a significant part of the national expenditures of any country and directly affects the competitiveness and trading capacity of a country in the world trade system. This fact can be seen by the growing participation of the Brazilian economy through exports and purchasing power per capita, and the increment of transportation as a percentage of participation in the Gross Domestic Product, rising from 3.7% to 6.9% between 1985 and 2019[1].

Several factors impact the efficiency of road freight transportation in Brazil. The most alarming one being the overburdened road system when compared to other countries of continental dimensions, such as USA and Canada. According to data from the National Confederation of Transportation (CNT), approximately 59% of freight transportation in Brazil (and 93 % in the State of São Paulo) is moved through the road system[2]. Additionally, the conditions of the national fleet and roads, on average, are far below those of the aforementioned countries. These facts create a difficult environment to manage transportation operations efficiently at a national level (Ribeiro and Ferreira, 2002).

Apart from the inadequate conditions of the physical resources involved in transportation, there are other factors that impact freight transportation operations. Such is the case of the emergence of time-sensitive paradigms for planning and executing supply chain operations (e.g. Just-In-Time and Quick-Response-Manufacturing) whose concern for reducing lead times tightens the flexibility for carriers to maneuver their resources in moving products among different locations (Chase et al., 1998). As a result of these changes, carriers have to rely on information technologies to enhance their response capabilities to customers and stay competitive in this evolving market. Among those technologies that

[1]Source: https://data.oecd.org/brazil.htm#profile-economy; IBGE, Diretoria de Pesquisas, Coordenação de Contas Nacionais http ://www.ibge.gov.br Accessed: 21-04-2020

[2]Source: FIESP, http://www.fiesp.com.br/transporte-e-logistica/matriz-de-transporte/ Accessed: 30-07-2021

have revolutionized the sector, are: Electronic Data Interchange (EDI) and the internet, which speed up the exchange of information between organizations and customers; Global Positional Systems (GPS), which allows to control and track on real-time the movement of vehicles; and Decision Support Systems (DSS), which supports a more robust decision making process for the organization (Crainic, 2003; Roy, 2001).

Considering the challenges faced by carriers where inherently complex transportation operations have to be planned and executed in uneasy environments as a result of the aspects mentioned above, this work aims at contributing with optimization tools, which serves as a basis for developing DSS, to support decisions faced by road freight carriers in managing their fleet. As a consequence of different demand levels for transportation services among geographically dispersed locations, it is common for vehicles to accumulate in some regions whereas falling short in some others where they are indeed needed. Even though the repositioning of empty vehicles do not contribute to the profits of carriers (their operational costs have to be covered by loaded trips), these movements are essential for providing continuity to the overall operations of freight transportation. Hence, repositioning empty resources like vehicles constitutes an important component in the planning activities of carriers. In this context, the Vehicle Allocation Problem (VAP) comes out as a tool for supporting these decisions, and consists in: given a set of demands for freight transportation services among different terminals, how to reposition vehicles so as to serve these demands, maximize the generated profits and minimize the cost of empty trips.

1.1 Objectives and Contributions

Previous works that treated the VAP in solving large realistic-size instances faced by Brazilian freight carriers resorted to heuristic and metaheuristic techniques as the computational limitations of general-purpose solvers did not allow to process these instances (Vasco and Morabito, 2016a,b). Although this approach is pragmatic as it yields feasible solutions for the problem at hand, it poses the predicament to the decision maker of not being able to evaluate how good are the solutions to be implemented. Hence, the objective of this work is to study the VAP and to develop modelling approaches and exact solution methods that provide optimal solutions or, at least, quality certificates of the solutions for large-scale realistic-size instances. The main contributions resulting from this work are: a Branch-and-Benders-Cut (BBC) for solving the VAP, a Branch-and-Price (BP) method for solving large-scale instances of the VAP, and finally, a new formulation, the node-demand formulation, based on sequencing requests for solving the VAP, and a BP method for solving this new formulation.

1.2 Organization

The remainder of this text is organized as follows: Chapter 2 presents a brief description of the freight transportation operations and introduces the VAP, its deterministic mathematical formulation and a toy-problem to illustrate the problem and facilitate its understanding. Chapter 3 presents the literature review of the problem being studied. Chapter 4 presents the BBC for the arc-demand formulation of VAP. Chapter 5 presents a BP method for solving the arc-demand formulation of the VAP. Chapter 6 presents a new formulation of the VAP and a BP method for solving this new formulation. Finally, Chapter 7 presents the conclusions of this work and some perspectives for future research.

Chapter 2

Freight Transportation

In this chapter, we briefly describe freight transportation activities, as well as introduce the Vehicle Allocation Problem (VAP).

2.1 Description of Freight Transportation Activities

Freight transportation is an essential component of any economy as it enables to shorten distances between geographically separated supply and demand locations. Freight transportation can be divided into five categories or modes, each one presenting distinct advantages, depending on the characteristics of the load and the service required. The modes are: railway, roadway, maritime, pipeline and airway. Table 2.1 shows a comparison of these modes across five characteristics, being five the most favourable and one otherwise (Ribeiro and Ferreira, 2002).

Characteristics	Railway	Roadway	Maritime	Pipeline	Airway
Velocity	3	2	4	5	1
Availability	2	1	4	5	3
Reliability	3	2	4	1	5
Capacity	2	3	1	5	4
Frequency	4	2	5	1	3

Table 2.1: Operational Characteristics of Transportation Modes. Source: Ribeiro and Ferreira (2002).

Although each mode bears distinct advantages for each type of freight, Table 2.2 shows how unbalanced is the Brazilian transportation mode matrix, favouring the roadway system. With high logistics costs incurred, as a consequence of the fewer options and hence inadequate modes of transportation in some regions, the internal competitiveness is upset in the Brazilian market, especially within the less developed regions (North and North-East). According to data from the Logistics and Transportation National Plan (PNLT), the avoidable logistics costs (either in the internal or external market) are in the order of US$2.5 billions per year. In view of the high participation of transportation costs within

the overall logistics costs (approximately 31.8%), a more balanced transportation matrix would result in a significant cost reduction and a benefit for the national economy [1]. Furthermore, considering that changes directed to balance the transportation matrix involves long-term planning of different actors, it is important to develop tools that support an effective decision-making process of transportation operations on the already overburdened road system.

Freight Transportation Matrix		
Mode	**Millions (Tons per kilometer)**	**Proportion (%)**
Roadway	485.625	61.10%
Railway	164.809	20.70%
Maritime	108	13.60%
Pipeline	33.3	4.20%
Airway	3.169	0.40%
Total	794.903	100%

Table 2.2: Freight Transportation Matrix. Source: Serrano Colavite and Konishi (2015).

The freight road transportation system can be divided in two categories. In long-haul or large-scale freight transportation, goods are moved over long distances, the time-span of the involving activities (trips, loading and unloading at terminals, among others) are long and loads (goods aggregated to a single unit in order to be transported) are high-volume and big-weight, which limits their handling at certain facilities and places. In short-haul or short-scale freight transportation, the services have a limited coverage in terms of distances and times. Given the smaller loads (single boxes or single products), it is easier to interact closer to the consumer (Crainic, 2003). In this work, the long-haul freight transportation is the one to be approached.

Long-haul freight transportation systems can be divided in two subcategories, depending on the entity transporting the loads: the shipper (producer) is the same carrier (the shipper owns its own fleet and other resources necessary for moving loads) or the transporter can be a logistics operator, a "for hire" carrier. In the former, there exists a limited number of origins that correspond to the company's facilities and a varied number of destinations. In this globalized environment, companies frequently prefer to concentrate their efforts in their core activities and turn to the latter, logistics operators, to perform their transportation activities. Logistics operators serve the transportation demands of several companies at the same time, with different points of origin-destination, different types of loads, thus turning their activities more complex when compared to the transportation activities of shippers with their own fleet. In this work, the logistics operator decisions are the ones to be treated.

Logistics operators usually offer two types of services: customized services and consolidation services. In customized transportation, such as Truckload Trucking (TL Trucking), carriers offer a typical door-to-door long distance transportation. In this mode, each vehicle (or convoy of vehicles) is dedi-

[1] Source: Plano Nacional de Logística e Transportes Accessed: 21-04-2020.

cated exclusively to one customer. When a customer calls for a service, a driver/truck is assigned to it, and the vehicle is deadhead to the customers location to be loaded. Then, the vehicle moves to the destination; thus performing the long-haul transportation part of the operation. When the vehicle is unloaded, the driver informs its current state (location, loading state) to the central planner or dispatcher who decides if the vehicle waits for another demand to be served at the same location or move to another more strategical location to anticipate a possible future demand.

Sometimes, transportation services cannot be customized because the loads are too small to call for a full truckload service, or even it would be too expensive to hire smaller vehicles for that cause. In that case, carriers provide consolidation of loads to serve customers while taking advantage of the economies of scale provided by a truckload. One such service that works on the road system is the Less-than-Truckload Trucking (LTL Trucking). When demands of several customers are served by a common vehicle or convoy of vehicles, services cannot be tailored individually for each customer. Therefore, carriers rely on a set of regular services and a network of hubs that supports those services. Hubs are distribution centres or cross-docking facilities whose objective is to consolidate and reorganize the goods into loads. Based on the portfolio of customers, the carriers establishes a set of schedules or rules, and routes between hubs that aim at fulfilling the demands effectively. Figure 2.1 shows a typical flow of operations for a LTL trucking operator. The cycle begins by collecting some goods that need to be transported, which are transferred to the information processing centre. By means of a Decision Support System (DSS) or based on the experience of the dispatcher, a vehicle picks-up the load and takes it to a end-of-line terminal or hub, where they are unloaded and related information and documents are verified (Bill of Lading): weight, dimensions, type of freight, quantity, among others. Then, the load is classified according to its immediate destination and loaded in line-haul trailers, or simply moved to a nearby break-bulk terminal in order to be classified and consolidated. At this point begins the long-haul part of the transportation where trucks are used to move big loads along distances (Roy, 2001).

Trailers are routed across intermediary terminals with consolidated (grouped) loads (possibly from different origins) heading to a common destination. At this point, loads can be again unloaded, classified and loaded again. Sometimes goods can be kept in the trailer while other merchandise is added to the trailer without needing to reorganize, which reduces the handling costs and operations time. The number of stop-overs at terminals depends on the level of service required by the customer. Finally, at the destination (end-of-line) terminal, the load is unloaded, verified, classified, codified and moved to the docks where smaller vehicles will deliver them. It is customary to transport the line-haul shipments among terminals at night, so as to make deliveries at the beginning of each day.

One of the major problems faced by the carriers in both cases (TL and LTL Trucking) is what to do with the vehicles after they make the deliveries. In the USA, for instance, empty vehicle movements

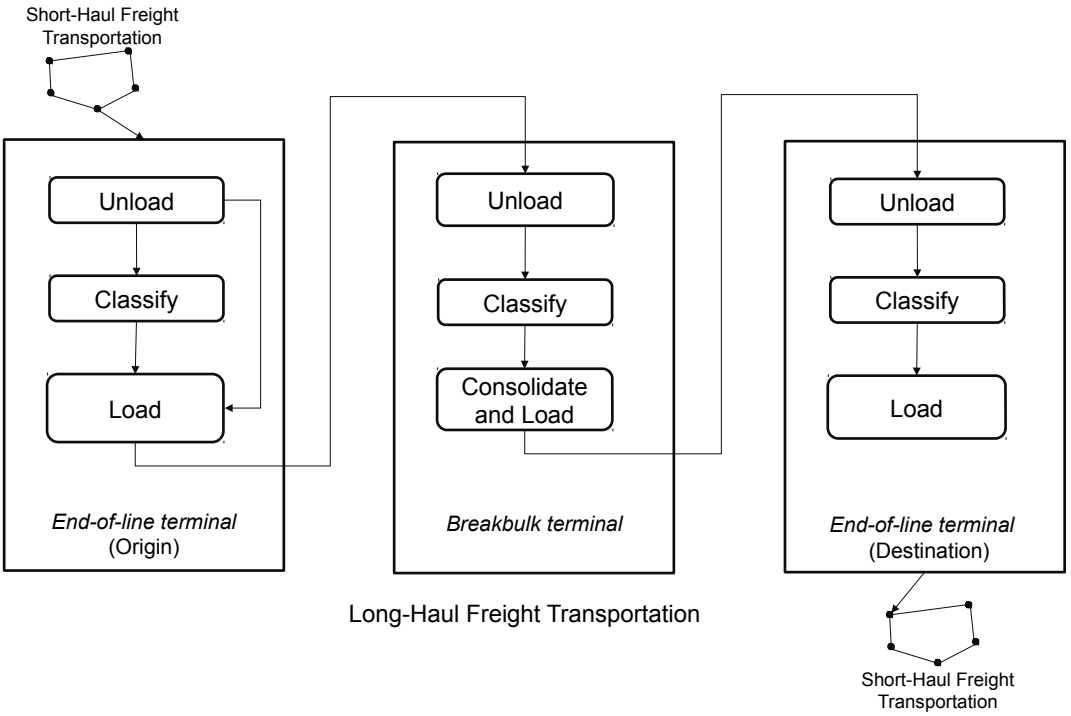


Figure 2.1: Graphic representation of activities of *LTL Trucking* operator.

accounts for 18% of daily movement, which evidence a high proportion of movements that do not contribute to profits directly and impact the environment in a negative manner (Liu et al., 2010a). However, due to the geographical dispersion and demand uncertainty along the planning horizon of the carrier, it is necessary to reposition these resources in order to reduce the opportunity cost of losing possible customers. In this line, there exists the necessity to develop tools to improve the decision-making process of this kind.

2.2 Vehicle Allocation Problem

In this context, arises the VAP, which can be briefly described as follows. A shipper call a carrier to move a load from location A to location B. The carrier must deadhead a truck to the shipper where the trailer is loaded and taken to location B. Once delivered, the carrier must decide what to do with the truck at the arrival location. When routing the vehicle, more loads can be requested. Therefore, at any point in time, the truck must be assigned another load, repositioned empty to another location in anticipation of forecast loads, or held in its current location. It is important to note that there is no consolidation of freight, as in vehicle routing or less-than-truckload routing, so that efforts can be put forth to efficiently allocate vehicles (resources) to loads (tasks) over time.

Two important dimensions arise from this example that have to be considered for treating real-life problems. First, the evolution of information, relates to the fact that in some problems the information

available to the carrier may change during the execution of the plan, for example, the customer may change the delivery location or loads may be called in after the execution of the plan. Second, the quality of information relates to the level of certainty on the available data, for example, when customer's demand or travel times are only known as a range estimate. Based on these two dimensions, Pillac et al. (2013) present a categorization for routing problems as shown in Table 2.3.

Table 2.3: Taxonomy of vehicle routing problems based on information evolution and quality.

		Information Quality	
		Deterministic Input	Stochastic Input
Information Evolution	Input known beforehand	Static and Deterministic	Static and Stochastic
	Input changes over time	Deterministic and Dynamic	Dynamic and Stochastic

Source: (Pillac et al., 2013).

In static and deterministic problems, all input is assumed to be known beforehand and routes or decisions are not changed during the execution of the plan. In static and stochastic problems, the input parameters are known as random variables, minor recourse actions are considered for different realizations of the values and routes or decisions are not changed during the execution of the plan. In deterministic and dynamic problems, part or all of the the input is unknown and revealed dynamically during the execution of the plan. Finally, in dynamic and stochastic problems, part or all of the input is unknown and revealed dynamically during the execution plan, however, contrary to the later category, there is some probabilistic knowledge on the unknown data. The last two categories, the dynamic problems, represent an important trend for treating real life problems since technological support for real-time communication (Geographical Positioning System - GPS and Geographical Information System - GIS) between the environment (changing attributes of the vehicles, conditions of the networks, customers, among others) and the decision maker is more accesible nowadays, and thus can be used as a tool to increase competitiveness for the organization (Marchet et al., 2012; Roy, 2001).

The problem treated in this work was initially presented as the Dynamic Vehicle Allocation Problem in the work of (Powell, 1986), where the identifier dynamic was due to the fact that decisions are staged over time. However, given this rather simple definition, it is possible to define a dynamic model of a problem in a static manner as shown in Aronson (1989) and Rockafellar (1998). That is, a dynamic network construct can be defined as a graph where each node is a pair location-time and each arc represents a decision that extends over several periods, however, uncertainty and changing input data is not incorporated into the model. Therefore, the conflict over the word dynamic arises when applied to problems and models, and that is an important distinction to make (Powell et al., 1995). A problem is dynamic if one or more of its parameters is a function of time. This includes problems with time-windows and varying travel times. The first type fits into time-dependent data problems where data is known in advance. The second type fits into dynamic data problems where data changes constantly

over time.

Similarly, a model is considered dynamic when interaction between activities are explicitly incorporated over time, as in the work of Aronson (1989). Within dynamic models, it is important to distinguish between deterministic dynamic models and stochastic models which truly captures the staging of decisions and the realization of random variables. In fact, many deterministic dynamic models are solved without considering the time-structured format of the model. By contrast, when solving stochastic dynamic models, specific steps for approaching the time-structure need to be taken into account in order to design a solution strategy.

Finally, there is also the application of a model. In this context, the application of a model is dynamic if it needs to be solved repeatedly or in an on-line fashion as new information is received. Hence, based on the characterization of the definition of dynamism of Powell et al. (1995), the taxonomy of vehicle routing presented in Pillac et al. (2013) refer to dynamism in the context of application of the model. In order to keep up with the trends in the using of word dynamic, we decided to drop the identifier dynamic from the problem's name as opposed to Vasco and Morabito (2016b).

2.3 An integer linear programming model for the VAP

In this section, we present a integer linear programming (ILP) model for the VAP, which is an extension of an Integer Multicommodity Network Flow Problem. We call this formulation the arc-demand formulation, where demand is represented by arcs in a graph, as opposed to the formulation proposed in Chapter 6 where demand occurs at the nodes of a graph. Let N be the set of terminals; T the set of time periods; and V be the set of types of vehicles composing the fleet. Additionally, consider the following parameters:

- τ_{ij} : travel time from terminal i to terminal j, $\forall i, j \in N$.
- d_{ijt} : demand for transportation services (number of loaded vehicles) from i to j beginning at time t, $\forall i, j \in N$, $\forall t \in T$.
- p_{ijv} : profit (income minus direct operational costs) obtained by serving the route from terminal i to terminal j with a vehicle of type v, $\forall i, j \in N$, $\forall v \in V$.
- c_{ijv} : cost of moving an empty vehicle of type v from terminal i to terminal j, $\forall i, j \in N$, $\forall v \in V$.
- m_{itv} : quantity of vehicles of type v that enter (i.e., become available) the system at terminal i at time t, $\forall i \in N$, $\forall t \in T$, $\forall v \in V$.
- A_{ijv} : restriction of movement between terminals i and j for vehicle type v, being 1: if the vehicle is allowed to move, and 0: otherwise, $\forall i \in N$, $j \in N$ and $v \in V$.

Let the decision variables be:

- x_{ijtv} : flow (number) of loaded vehicles of type $v \in V$ that start moving from terminal i to terminal j at time t to satisfy demand d_{ijt}, $\forall i \in N$, $j \in N$ and $t \in T$.
- y_{ijtv} : flow (number) of empty vehicles of type $v \in V$, that start moving from terminal i to terminal j at time t, $\forall i \in N$, $j \in N$ and $t \in T$.

From the above definitions, the model writes as:

$$\max \quad \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} (p_{ijv} x_{ijtv} - c_{ijv} y_{ijtv}) \tag{2.1}$$

$$\text{s.t:} \quad \sum_{j \in N} (x_{ijtv} + y_{ijtv}) - \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \left(x_{ji(t-\tau_{ji})v} + y_{ji(t-\tau_{ki})v} \right) - y_{ii(t-1)v} = m_{itv}, \tag{2.2}$$

$$\forall i \in N, \forall t \in T, \forall v \in V,$$

$$\sum_{v \in V} x_{ijtv} \leq d_{ijt}, \qquad \forall i, j \in N, \forall t \in T, \tag{2.3}$$

$$x_{ijtv} = 0 \wedge y_{ijtv} = 0, \ if \ A_{ijv} = 0, \quad \forall i, j \in N, \forall t \in T, \forall v \in V, \tag{2.4}$$

$$x_{ijtv} \in \mathbb{Z}_+, y_{ijtv} \in \mathbb{Z}_+, \qquad \forall i, j \in N, \forall t \in T, \forall v \in V. \tag{2.5}$$

The objective function (2.1) maximizes the total profit over the planning horizon, which is equal to the income generated from the loaded vehicle trips minus the cost of the empty vehicle trips. As shown in Figure 2.2, constraints (2.2) guarantee the flow of vehicles at each terminal i at time t for each type of vehicle v. Constraints (2.3) establish an upper bound to loaded trips between terminals, which equals the demand on that route. Constraints (2.4) establish the trips (loaded or empty) that can not be made by each type of vehicle. This constraints (2.5) impose restrictions on the variable's domain. It is worth mentioning that demand is defined in number of vehicles and all vehicle types are assumed to have the same capacity, hence, the factor of conversion between loads for each vehicle type to fulfilled demand is one in constraints (2.3). Nonetheless, the fleet is considered heterogeneous as owned and hired vehicles generates different costs and profits (objective function (2.1)) as well as certain trips are restricted for different types of vehicles (constraints (2.4)).

Note that $m_{itv} = -1$ can have the meaning of vehicles leaving the system during the planning horizon due to scheduled fleet maintenance (and/or repairs) or finished activities of outsourced fleet.

In some practical situations it is desirable to treat each vehicle individually instead of grouping them into types. This practice is purposed to improve the planning and control of the operational resources

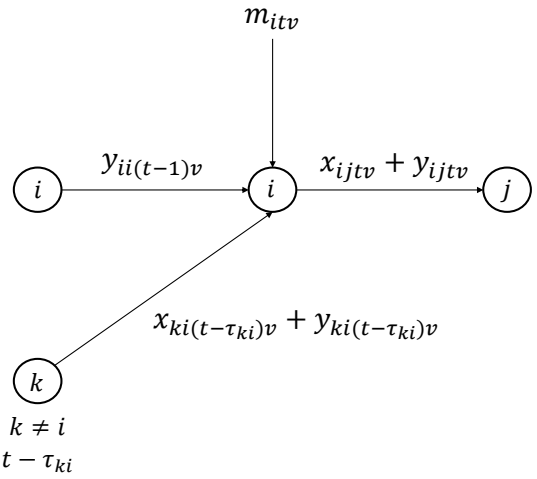


Figure 2.2: Graphic representation of flow conservation.

(Vasco and Morabito, 2014). Hence, each individual vehicle is considered as an individual type of vehicle and the supply of vehicle turns into a binary parameter, i.e., $m_{itv} \in \{0, 1\}, \forall i \in N, \forall t \in T, \forall v \in V$, such that

$$\sum_{i \in N} \sum_{t \in T} \sum_{v \in V} m_{itv} = |V|.$$

When considering each vehicle individually, the size of the ILP model grows drastically as a function of the number of variables and restrictions, thus, making it harder to be solved through general purpose optimization solvers.

An important point to make is that when the fleet is homogeneous, the index v is not necessary anymore and constraints (2.3) turn into an upper bound for an integer variable. The resulting model of the homogeneous fleet is a simple extension of a minimum cost flow problem for which exact polynomial algorithms exists. Nonetheless, the problem we are solving in this work contemplate different profits, costs and restriction of movements on the arcs for each vehicle type, which makes it an extension of the multicommodity cost flow problem for which there is no polynomial algorithm (Ahuja et al., 1993).

This model can be extended to represent situations in which vehicles need to disappear from the network during the planning horizon. These events can be due to scheduled maintenance or finishing activities by outsourced vehicles, among others. The extension can be done by allowing m_{itv} to take negative values, thus representing the quantity of vehicles of type $v \in V$ that exit the system at terminal $i \in N$ in period $t \in T$.

2.3.1 Illustrative Example

To facilitate the understanding of the mathematical model for representing the problem, the following is a small-scale example.

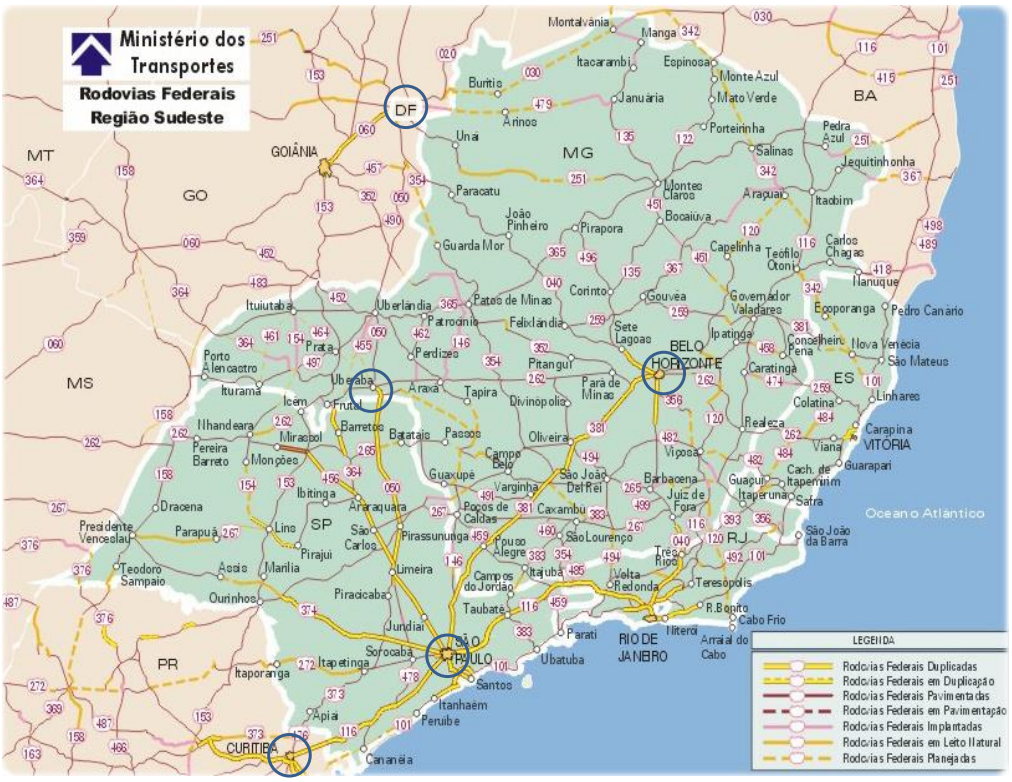


Figure 2.3: South-east Brazil.

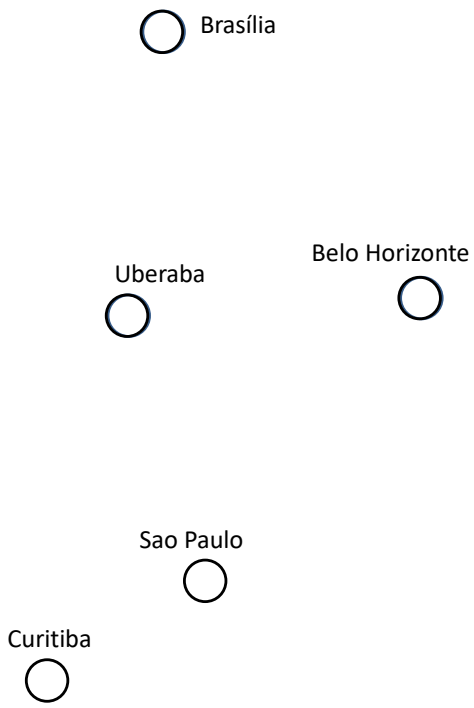


Figure 2.4: Set of terminals.

TransBras is a TL trucking logistics operator acting in the south-east region of Brazil, see Figures 2.3 and 2.4. Four requests for transportation of full loads were made: from Brasilia to Belo Horizonte one load the first day, from Sao Paulo to Belo Horizonte three loads the second day, from Uberaba to Sao Paulo one load the fourth day and from Curitiba to Brasilia one load the fifth day (Figure 2.5). There are two available type-1 trucks the first day in Curitiba and there will be two type-2 trucks available the second day in Uberaba. Travel times between the cities are estimated based on an average speed and are presented in Table 2.4. Profit for servicing each demand and costs incurred from moving empty truck are presented in Tables 2.5 and 2.6. Let $T = \{1, ..., 4\}$ be the set of periods (days) and $N = \{$Brasilia, Belo Horizonte, Uberaba, Sao Paulo, Curitiba$\} = \{1, ..., 5\}$ be the set of terminals. Then, the demand is represented as: $d_{121} = 1, d_{422} = 3, d_{344} = 1, d_{515} = 1$.

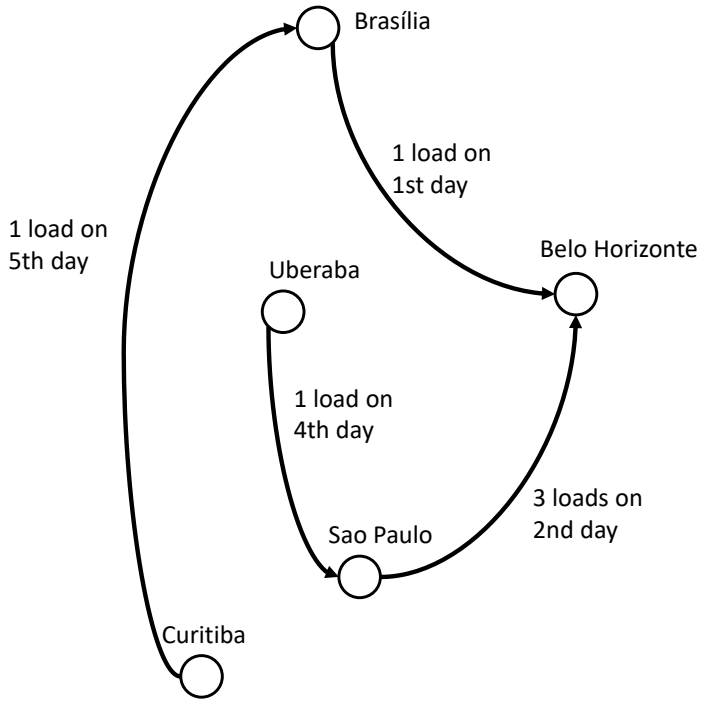


Figure 2.5: Problem's parameters.

Table 2.4: Travel times between terminals for the TransBras example.

τ_{ij}	1	2	3	4	5
1	0	1	2	1	3
2	1	0	1	2	2
3	2	1	0	1	2
4	1	2	1	0	1
5	3	2	2	1	0

There are two types of vehicles, i.e., $|V| = 2$. In addition, trips between Brasilia and Belo Horizonte are not allowed due to road maintenance, i.e., $A_{121} = A_{211} = A_{122} = A_{212} = 0$. Figure 2.6 shows a

Table 2.5: Cost of empty vehicle movement for the TransBras example.

c_{ijv}	1,1	2,1	3,1	4,1	5,1	1,2	2,2	3,2	4,2	5,2
1	0	1	2	2	2	0	3	3	2	2
2	1	0	2	2	2	3	0	3	3	2
3	2	2	0	2	1	3	3	0	1	2
4	2	2	2	0	1	2	3	1	0	3
5	2	2	1	1	0	2	2	2	3	0

Table 2.6: Profits for the TransBras example.

p_{ijv}	1.1	2.1	3.1	4.1	5.1	1.2	2.2	3.2	4.2	5.2
1	0	1.8	3.6	3.6	3.6	0	4.2	4.2	3.6	3.6
2	1.8	0	3.6	3.6	3.6	4.2	0	4.2	4.2	3.6
3	3.6	3.6	0	3.6	1.8	4.2	4.2	0	4.5	3.6
4	3.6	3.6	3.6	0	3.6	3.6	4.2	4.5	0	4.2
5	3.6	3.6	1.8	3.6	0	3.6	3.6	3.6	4.2	0

graphic representation of the problem's parameters over an extended space-time network. The bold and small arrows represent arcs were there are demand for loaded trips and supply of vehicles at terminals, respectively. The optimal solution for the VAP is presented in Figure 2.7. The solid arrows represent full-load vehicle movements, for instance, 2 type-1 loaded vehicles travel between terminals 4 and 2 starting at period 2. The dash arrows represent empty vehicle movements or vehicles held idle in the same terminal, for instance, 2 type I empty vehicles travel between terminals 5 and 4 starting at period 1 while 2 type-2 vehicles are held idle for 1 period starting at terminal 2 and period 2.

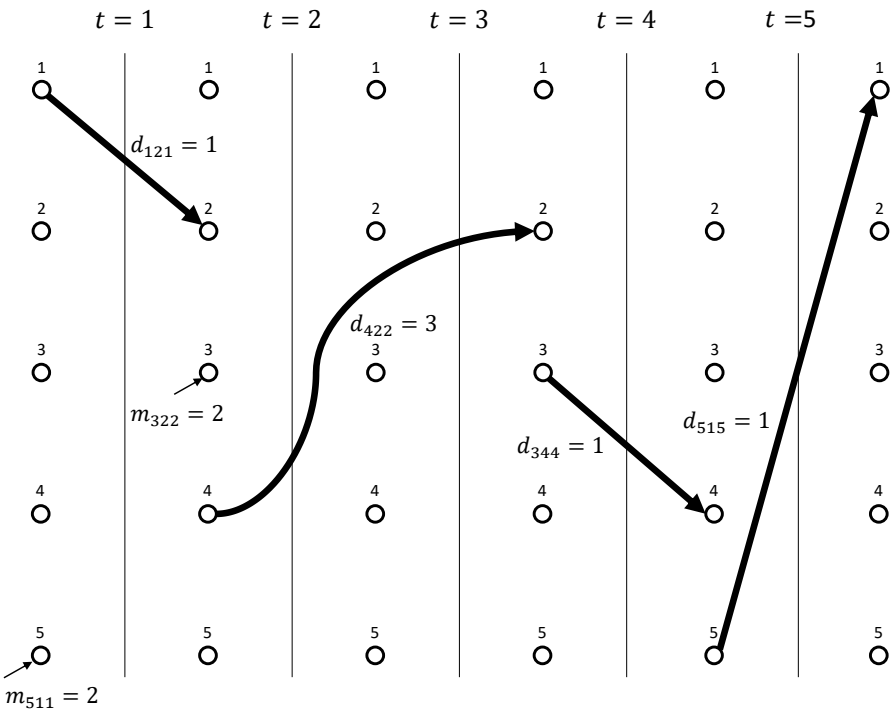


Figure 2.6: Problem's parameters in the space-time network.

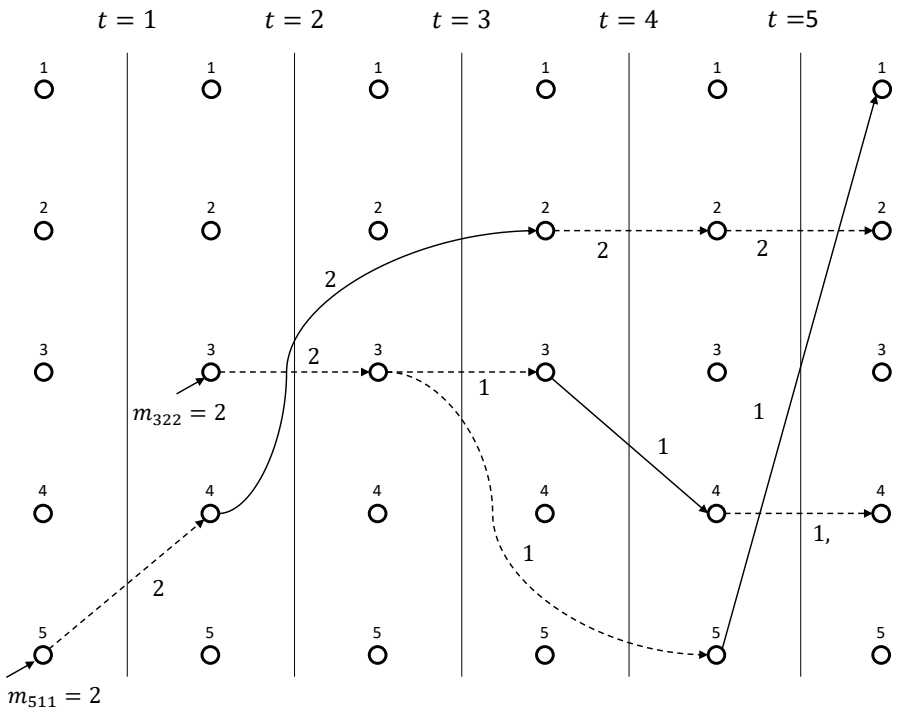


Figure 2.7: Optimal solution for the TransBras example.

Chapter 3

Literature Review

This section presents a literature review of works related to allocation of vehicles that serve freight transportation. We first describe two modelling approaches for dealing with the present problem within the mathematical programming paradigm: network flow formulations and vehicle routing formulations. In addition, we present works on related problems that use decomposition methods.

3.1 Arc-demand formulations

This section describes the deterministic and stochastic models that were developed for the problem of allocating vehicles based on the arc-demand formulation.

Powell et al. (1984) use a time-space expanded graph to represent the time-dependent aspect of the decision making process in the allocation of empty vehicles. Given the demand is random, supply of vehicles beyond the first time period is also random. Therefore, the flow of vehicles corresponds to the fraction of the supply at each terminal to be sent, and that quantity is determined before the actual demand is realized. Given this assumption, the model generates empty miles by moving vehicles that are not loaded.

Powell (1986) proposes an alternative model that allows vehicles to be held in inventory from one period to another (null recourse) when the realized demand falls short from the outbound quantity of vehicles. The decisions to be taken correspond to: how many vehicles to send empty between terminals, how many vehicles should be held at a given terminal and how many vehicles should be allowed to handle demand between terminals. Given the structure of the constraints, the authors use the Frank-Wolfe algorithm for solving the problem.

Dejax and Crainic (1987) propose a taxonomy of empty repositioning problems, looking to increase the knowledge of these systems and identify possible trends for future research. The authors present a classification based on scope and coverage, which they consider pertinent for the study of empty

repositioning problems: operational models and policy-taking models.

Hughes et al. (1988) study the "end-effects" which are the result of considering a finite number of periods in the problem of allocating vehicles. Ideally, it would be desirable to consider infinite planning horizons; however, because of the resulting complexity of the problem, it is necessary to truncate the number of periods. The problem with truncation is that relevant information of the periods outside the planning horizon is not taken into account in the analysis, which can lead to suboptimal solutions in the global context of the operations. The authors present three methods to mitigate these effects: Dual Equilibrium, Generalized Summation e Leontieff Approximation. Although truncating the number of periods can have negative effects in long-term solutions, considering long periods when treating uncertain demands can also be counterproductive. According to Sethi and Sorger (1991), the main reason for considering finite planning horizons is that forecast of future distant demands tend to have low reliability. For this reason, the idea is to define a forecast horizon, which is the necessary number of periods in order to establish optimal policies for allocating vehicles without being affected by the remaining periods.

Frantzeskakis and Powell (1990) propose an N-stage stochastic programming formulation for the VAP. As in previous models, the travel times are deterministic one-period discrete parameters, the fleet is homogeneous and the demand for services beyond the current day of decision making is random. Loads that cannot be serviced are assumed to be lost. Given that partitioning methods that were developed for problems with fixed recourse would eliminate the network structure for this formulation that bears the property of network recourse, the authors propose a heuristic based on substitutions of the expectations by linear approximations for maintaining the desired property in order to use network techniques for solving the problem. The method is named Successive Linear Approximation Method (SLAM).

Cheung and Powell (1996) use the same multi-stage stochastic formulation of the aforementioned work and propose the SCAM method (Successive Convex Approximation Method). Based on the work of Powell and Cheung (1994), where they describe a method to find the exact expected function value for a tree with random arc capacities and deterministic supply of vehicles, they develop a backward recursion approach to successively calculate convex approximations of the expected recourse function in each stage.

Shi et al. (2014) propose a multi-stage stochastic programming model for treating the VAP with uncertain demands and customer chosen level of service. Given that deterministic travel times are parameters that define the form of the time-space network and varying travel times cannot be captured within this same network, the authors model the service levels as discrete random variables. Thus, an arc in the deterministic counterpart of the problem generates multiple arcs and arrival nodes in the

stochastic version depending on the possible realizations of the travel times at each stage. The authors propose a method called Successive Resource-Directive Decomposition Method (SREDM) which uses a backward recursion framework to successively provide convex approximations of the expected recourse functions at each stage.

Powell and Carvalho (1998) propose a formulation for treating the VAP based on logistics queueing networks (LQN). This formulation is based on discrete event dynamic systems, where demands are queued at terminals while waiting for available vehicles. In this formulation, the authors solve several sorting problems locally (for each terminal and each period), which consists in assigning units of vehicles to waiting customers (or demands). In order to make the local decisions to match the global optimization, two control mechanisms are proposed.

Erera et al. (2009) propose an adjustable robust formulation for repositioning empty resources on a time-space expanded network. The uncertain parameters in this model are the divergence of the nodes, which can represent the supply of or demand for empty resources. This approach aims at finding a repositioning plan that satisfies the flow bounds and balance constraints for the nominal values of the net supply of the nodes and is recoverable for each joint realization of the uncertain net supplies on each node. In order to limit overconservatism of the robust repositioning solution, the authors adopt the approach proposed in Bertsimas and Sim (2004) to restrict the potential joint realization of uncertain parameters using an uncertainty budget as a function of the number of parameters that can take the worst-case value. Three strategies for the repositioning model are proposed based on the recoverable actions.

Vasco (2012) and Vasco and Morabito (2016b) study the problem of fleet management in the context of freight transportation in Brazil. The authors part from a minimum cost flow-like model stated in Ghiani et al. (2004), and extend the model to include heterogeneous fleet, the necessity of outsourcing of vehicles, backlog of demand (given the high level of competitiveness is detrimental to reject services), capacity on the terminals, and restriction on the movement of vehicles through the networks. The resulting model is a deterministic multi-commodity network flow problem with inventory constraints for dealing with backlog. Because of the limitation of general purpose solver for solving the resulting models, heuristic and metaheuristics methods like GRASP, Simulated Annealing (SA) and Ant Colony Optimization (ACO) were used. Computational experiments were run on realistic instances of the problem. Cruz (2017) uses this deterministic model and propose an exact method based on Dantzig-Wolfe decomposition and Primal-dual Interior Point column generation for solving large-scale instances with a quality certificate for optimal solution. Computational experiments show that for instances where the fleet is totally disaggregated, this method gives better solutions in terms of efficiency and quality than the work of Vasco and Morabito (2016b).

3.2 Node-demand formulations

This section describes the main works within mathematical programming that treats the problem of allocating empty vehicles (or resources) to requests as a variant of a vehicle routing problem. The main idea is to route vehicles along the requests (demand for transportation service between two locations), while minimizing the movement of empty vehicles and complying with all requests, or maximizing the profits. In addition, in this literature there is the distinction between terminal and depot (one can be both if the depot has an associated request that needs to be serviced), and vehicles have to end their trips at depots exclusively.

Desrosiers et al. (1984) propose a model for treating the problem of shipping goods between a specific pair of terminals or customers. The problem is modelled as a VRP in which every vehicle has to depart and arrive at the same depot. It is assumed that vehicles have enough capacity for carrying the load, so that each demand between a pair of cities is represented as a node to be visited and vehicle's capacities are not considered. Also, each empty travel between the delivery of a load and the pickup of another load is represented by an arc. The objective is to minimize the empty travel costs while meeting the demands and respecting the time windows. They solve the problem using Branch-and-Bound (BB) and a set partitioning formulation using column generation. The subproblem for pricing feasible routes is a shortest path with time windows that is solved using an adaptation of Bellman-Ford algorithm, in which labels are two dimensional (arrival time, cost of route).

Desrosiers et al. (1988) propose a model for treating the problem of shipping goods between a specific pair of terminals or customers, and shipments constitute one or more full vehicle loads. It is assumed there are several depots and each vehicle has to depart and arrive at the same depot. The objective is to minimize the total distance travelled while meeting all the demand for full loads and respecting the duration of each trip to a prespecified time duration. The authors use an extended graph to reformulate the problem as an asymmetrical distance constrained TSP. The problem is solved by a BB algorithm in which subtour elimination constraints are initially relaxed.

Powell et al. (2000) propose an adaptive heuristic labelling algorithm for solving the assignment of trucks to loads. The algorithm is based on updating labels for loads and tasks in an online fashion. Labels of loads contain attributes that include: origin, destination, start of origin time window, end of origin time window and length of task. Labels for drivers contain attributes that include: location, time of availability, hours of service elapsed at any given time and daily duty time allowance. The model contains nonlinear restrictions and subtour elimination constraints. Thus, the authors propose two algorithms in which these restrictions are relaxed and the labels are updated by solving simpler problems (network flow problems) with penalization of the subtour constraints.

Arunapuram et al. (2003) propose a model for solving the multiple depot vehicle routing-scheduling problem with full truckloads. The objective is to minimize the travelled distance while meeting all demands for full loads with the vehicles available at each depot. The problem is formulated as an ILP with variables corresponding to feasible routes. The ILP is solved using an LP-based BB using column generation. Based on Desrochers and Soumis (1988), the authors proposed a DP algorithm for solving the pricing problem that takes into account time windows and waiting costs for generating feasible routes. Cuts are added to tighten the LP relaxation and branching procedures are based on the number of vehicles and the selected routes.

Li and Lu (2014) proposed an extension of the full truckload vehicle routing problem. In their problem, it is allowed that there are more than one delivery terminal for a single pick-up terminal, and one order is served several times by the same or different vehicles. The vehicles are required to satisfy: 1) the maximum load of each route does not exceed the vehicle's capacity, 2) the total travel time does not exceed the maximum duration and 3) the carrier vehicles are limited and the whole fleet departs and arrives at the same depot. Also, the outsourcing of fleet is allowed when necessary and profitable for the carrier. The objective is to maximize the profits of the logistics company. Since the problem has embedded a Chinese Postman Problem, it is NP-hard and the authors opt for a hybrid genetic algorithm for solving large scale instances.

Gendreau et al. (2015) study the one-commodity-full-truckload pickup-and-delivery problem, which is a especial case of the pickup-delivery vehicle routing problem in which the demand of a customer has to be delivered right after picking-up the corresponding load. It is assumed that the demand is unitary (one load per trip and vehicle) between pairs of terminals. The authors proposed two models that integrates a routing and an assignment problem in two fashions: an integrated nonlinear model and an integrated linear model. The structure of the integrated models lends themselves for using Benders decomposition and Generalized Benders Decomposition, respectively. Both reformulations are solved using Branch-and-Cut (BC) algorithms.

Gronalt et al. (2003) propose a model for the problem of pickup and delivery of full truckloads under time window constraints. The objective is to minimize the empty travelled distance while meeting all the demands and respecting the time windows for each order. The authors propose four heuristics: a saving algorithm, an opportunity saving algorithm, a simultaneous saving algorithm and the opportunity simultaneous saving algorithm.

Liu et al. (2010a) propose a model for treating a problem called the multi-depot capacitated arc routing problem with full truckloads. The objective is to determine the tours for a set of vehicles located at different depots to serve a set of orders and minimize the total shipping costs. The problem is NP-hard, hence the authors proposed a two-phased heuristic for finding feasible solutions. Liu et al. (2010b)

propose a model for an extension of the problem called the task selection and routing full truckload problem. In this problem, the carrier has to decide which services for full load transportation are served by private fleet and outsourced fleet. Each vehicle has to depart and arrive at the depot while respecting a given time span. It is assumed that each service can be served by only one vehicle, thus a service node (equivalent to a pair pick-up-delivery arc) can be visited only once. The objective is to minimize the fixed and variable costs incurred by the travel of each vehicle. A memetic algorithm is used to solve large instances of the problem.

Bai et al. (2015) propose a set covering model for solving the problem of transporting a large number of non-consolidatable commodities (containers) between a relatively small number of nodes (docks), satisfying time window constraints concerning commodities and drivers. The model is tailored to meet specific features, such as: shift-based schedules due to labor laws, consideration of service times at terminal since they are not so different from travel times, large variation of time windows for each commodity and a large number of commodities to handle in short periods over a small network of physical locations. The objective is to create a set of vehicle routes to deliver all commodities at minimum cost while respecting time windows. The authors propose a three-stage heuristic solution.

Coslovich et al. (2006) propose an ILP model for treating the problem of container management by the truckload carriers. The solution method consists in relaxing the constraints that couples the different decision in a Lagrangean fashion and obtain lower bound through subgradient optimization and upper bounds by construction heuristics.

3.2.1 Summary of the reviewed works and research development for this work.

Several works have been presented that solve the VAP or another related problem that consists in allocating empty resources to serve demand for road freight transportation services. Two main lines within mathematical programming were identified for dealing with this problem: network flow and fulltruckload formulations. There is another line of research based on dynamic programming, which is not discussed in the present work given its algorithmic approach differs from the one presented in this work. The following studies serve as a good starting point for the reader interested in this line of research: Godfrey and Powell (2001, 2002a,b); Powell et al. (2002); Spivey and Powell (2004). Table 3.1 summarizes the relevant characteristics of all works that are directly related to the VAP.

Table 3.1: Relevant characteristics for the current work on the VAP

Articles	Deterministic	Stochastic	Heuristic	Exact	Small-scale	Large-scale	Outsourcing
Powell (1986)		x		x	x		
Frantzeskakis and Powell (1990)		x	x			x	
Cheung and Powell (1996)		x	x			x	
Shi et al. (2014)		x	x			x	
Vasco and Morabito (2014, 2016b)	x		x			x	
Cruz (2017)	x		x			x	
Powell and Carvalho (1998)	x		x			x	
Erera et al. (2009)		x		x	x		
Powell et al. (2000)	x	x	x			x	
Desrosiers et al. (1984)	x			x	x		
Desrosiers et al. (1988)	x			x	x		
Arunapuram et al. (2003)	x			x	x		
Li and Lu (2014)	x		x			x	x
Gendreau et al. (2015)	x			x		x	
Gronalt et al. (2003)	x		x			x	
Liu et al. (2010a)	x		x			x	x
Bai et al. (2015)	x		x			x	
Coslovich et al. (2006)	x		x			x	

From the reviewed works of the previous sections, it can be noted that many studies of the space-time network and dynamic systems do not treat the multi-commodity case which is important and add complexity to the problem (Ahuja et al., 1993; Bertsekas, 1998). Other important aspect, is the lack of works addressing outsourcing possibilities in the network flow formulations. Finally, there is a gap in exact solution methods for solving large-scale instances of the deterministic network flow formulation of the VAP. The present work aims at filling these gaps by proposing exact solution methods for solving large-scale instances of the deterministic VAP considering the multi-commodity case. It is worth mentioning that exact decomposition-based methods have been used successfully to solve other network flow problems as stated in Section 3.3, hence the approach we take for developing our proposed method for the VAP. Furthermore, we propose a reformulation based on the node-demand literature and an exact method based on a BP for this new formulation. It should be noted that Gendreau et al. (2015) developed an exact solution method for solving large-scale instances of a similar empty repositioning problem. However, there is one aspect that differentiates their work from ours. The empty container repositioning problem they consider enables the possibility of a delivery terminal to be served from one among many pickup terminals. This is not the case for the problem we are currently dealing with, in which the pair pickup-delivery terminal is predefined.

3.3 Decomposition methods on related problems

This section describes the main works that use decomposition methods for solving related network flow-based logistics problems. The works reviewed in this section show promising results by applying decomposition methods to related NP-hard problems, namely the Multicommodity Network Flow Problem (MCFP) and the Capacitated Network Design Problem (CNDP). In many of these works, the decomposition has resulted in structural simpler subproblems such as the shortest path problem.

Jones et al. (1993) evaluate the impact of the type of formulation of the MCFP on the Dantzig-Wolfe decomposition. Three formulations are evaluated: several supply nodes to several demand nodes per commodity, one supply (demand) node to several demand (supply) nodes per commodity and one supply to one demand node per commodity. The authors conclude that, even though the compact formulation of the one-to-one formulation is inconvenient for large scale instances, it is more efficient when applied to the Dantzig-Wolfe decomposition.

Barnhart et al. (1994) uses a cycle-based representation to solve the MCNFP. This representation consists in describing solutions for each commodity as augmenting flow on cycles for a given initial path (see Flow Decomposition Theorem in Ahuja et al. (1993)). The authors use a constructive iterative procedure based on relaxing the domain of the flow on cycles in order to solve large scale instances found in telecommunications operations.

Barnhart et al. (2000) propose a Branch-and-Price-and-Cut (BPC) algorithm for solving the Origin-Destination Integer MCFP. The pricing problem is solved using shortest path algorithms. The branching procedure proposed is based on the variables of the reformulation and lifting cuts are added to break symmetry effects across the tree as well as tighten the linear relaxation.

Gondzio and Sarkissian (1996) use the Primal-Dual Column Generation Method (PDCGM) for solving the non-linear MCFP. Non-linear costs in network flow problems, as is this case, are used for modelling congestion effects. Contrary to the classical column generation where the master problem is solved exactly, this method begins the procedure with loose optimality tolerance, which is gradually adjusted as the optimal solution is approached. Similar to the Analytical Center Cutting Plane Method (ACCPM), this method takes advantage of the use of central prices without being computationally expensive as the ACCPM.

Holmberg and Yuan (2003) uses the column generation method to solve the MCFP with side constraints. The side constraints extension is suitable for modelling delay restrictions in data flow across telecommunication networks. These side constraints are handled within the pricing problem, which turn out to be a constrained shortest path problem. The pricing problem is solved using a pareto-based multiple label-correcting algorithm.

Peeters and Kroon (2008) describe a model and a Branch-and-Price (BP) method for solving the rolling stock circulation problem. This problem consists in assigning train units to one or several railway lines, constrained to predetermined timetables and technological aspects of the operations, in order to minimize seat shortages (hence maximize the level of service provided). The modelling is based on transition graphs for each train. The pricing problem resulting from applying the Dantzig-Wolfe (DW) decomposition is a shortest-path problem. The authors used an adaptation of the branching procedure described in Barnhart et al. (2000).

Alvelos and Carvalho (2007) present an extended model and a column generation method for the origin-destination MCFP. This new model has additional variables associated to flow on circuits and additional constraints to ensure equivalency to the classical formulation of the MCFP. The extra variables are exclusively considered in the restricted master problem, hence, the pricing problem can be solved by shortest path algorithms.

Moccia et al. (2009) studies the dynamic generalized assignment problem, which is an extension of the assignment problem over a time discrete graph, with warehouse and yard management related constraints. The authors present three formulations and show that the strongest one models the problem as a single origin-destination integer MCNFP. The authors use a column generation embedded in a heuristic to find lower and upper bounds. The pricing problem reduces to a shortest-path problem called the generalized cardinality-constrained shortest path problem on a layered graph (GCSPPLG), which is shown to be polynomially solvable. The structural properties of this problem relevant for dynamic programming algorithms was also studied in Spivey and Powell (2004).

Gendron and Larose (2014) present a Branch-and-Price-and-Cut (BPC) algorithm for solving the related capacitated fixed-charge network design problem. The restricted master problem is obtained by considering a subset of commodities of the arc-node compact formulation. The subproblem, which prices out arc-flow variables, is solved through the lagrangian relaxation of the flow conservation and capacity constraints. Cut generation with strong inequalities is used to strengthen the linear relaxation and, when needed, reliability branching (see Achterberg et al. (2005)) on the fixed-charge binary variables is applied to obtain a integer optimal solution.

Sridhar and Park (2000) propose a Benders-and-cut algorithm for the fixed-charge CNDP, which incorporates Benders and cutset inequalities into the BB. The authors test the algorithm on a range of problems with different traffic loads (proportion of total demand to the capacity of the network) and show that Benders cuts are more effective under heavy traffic load, while cutset inequalities are more effective under light traffic load. There is an extensive review of Benders decomposition applied to this problem in Costa (2005).

Costa et al. (2009) study the Benders decomposition on the related multicommodity CNDP. The authors show the relations between three types of inequalities: Benders, metric and cutset. In addition, they highlight the importance of strengthening Benders and cutset into metric inequalities for computational efficiency gains.

Lee et al. (2013) propose a Benders decomposition for solving the multicommodity CNDP with demand uncertainty. By fixing the design variables, the Benders subproblem results in a multicommodity network flow problem. Uncertainty in demand (weights in flow variables) is addressed through the polyhedral uncertainty set by Bertsimas and Sim (2004), which only affects the Benders subproblem. They propose a simultaneous cut generation to accelerate the algorithms convergence.

Chapter 4

A Branch-and-Benders-Cut method for the VAP

In this chapter, we explore the Benders decomposition method as applied to the VAP. More specifically, we propose a Branch-and-Benders-Cut (BBC) which consists of adding Benders cuts whenever an integer solution is found along the Branch-and-Cut (BC) tree of a general purpose solver via lazy constraints. We compare the results from this method to the ones obtained by solving the compact formulation of the VAP using the standalone BC of the solver and the automatic Benders decomposition strategy of the solver.

4.1 Benders Decomposition for the VAP

Benders decomposition is suited for problems whose variables can be partitioned into two sets, hence, the set of decisions can be separated according to a trial-and-error-like method of two different stages. This method works as follows. Suppose the company described in Section 2.3.1 has two departments. The sales department (A) is in charge of taking requests for freight transportation services and deciding which to attend, while the operations department (B) is in charge of allocating empty vehicles to comply with the sales department plan. Suppose this week the sales department received the requests illustrated in Figure 4.1.a (requests are represented by bold arcs between two terminals while small short arcs represent the supplied quantity of vehicles at a given terminal), and naturally they accepted all requests as specified in Figure 4.2.b, wherein all x variables were set to match the demand d.

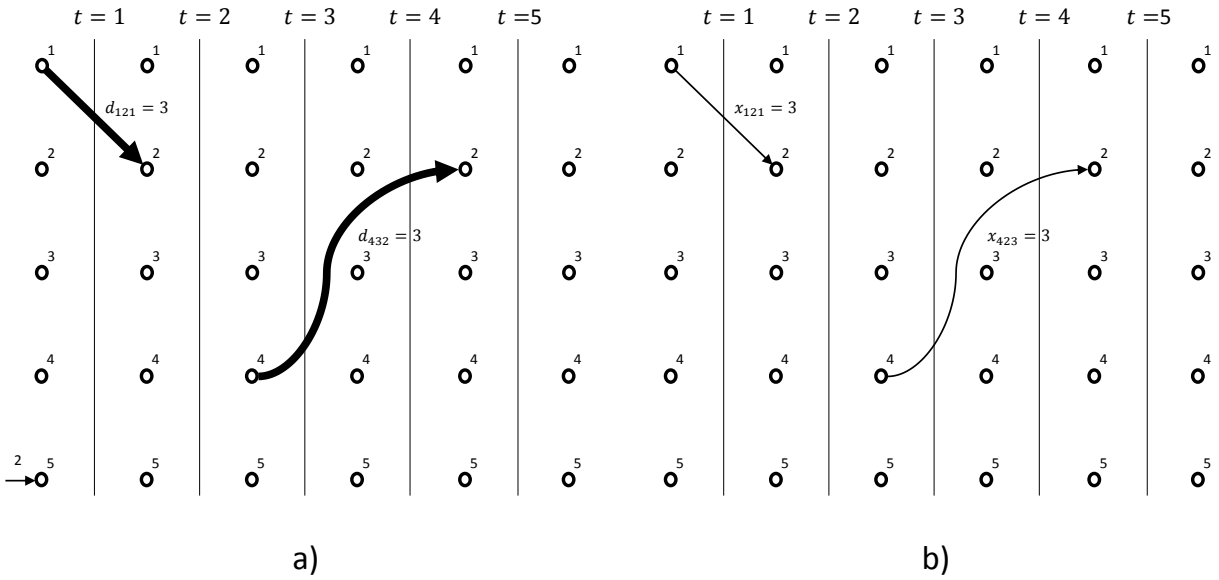


Figure 4.1: a) Parameters. b) First x decision.

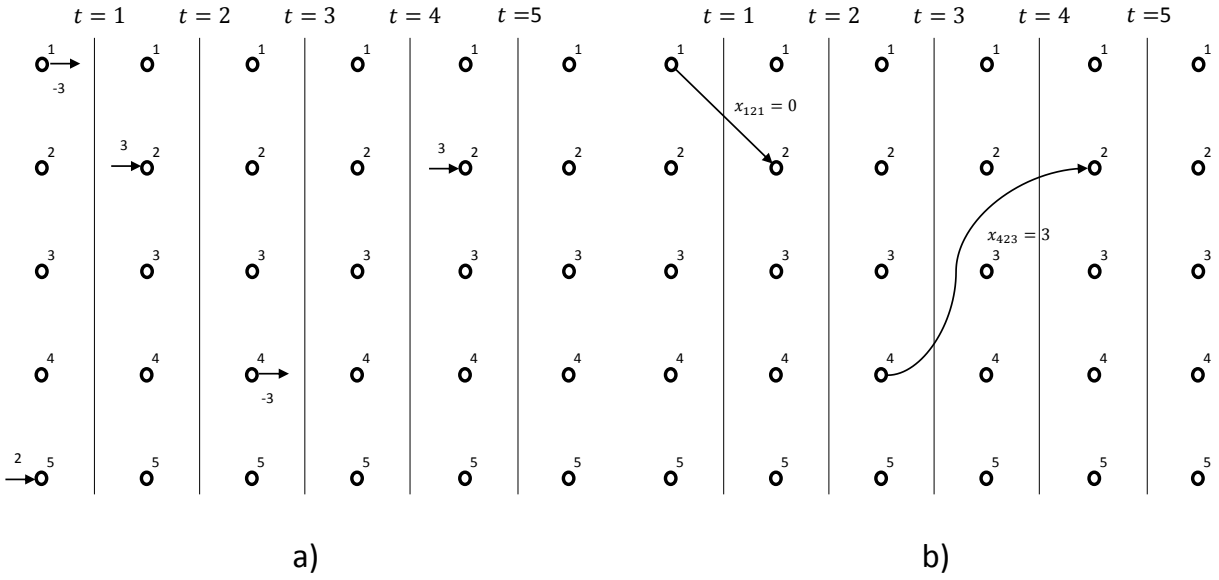


Figure 4.2: a) First y decision. b) Second x decision.

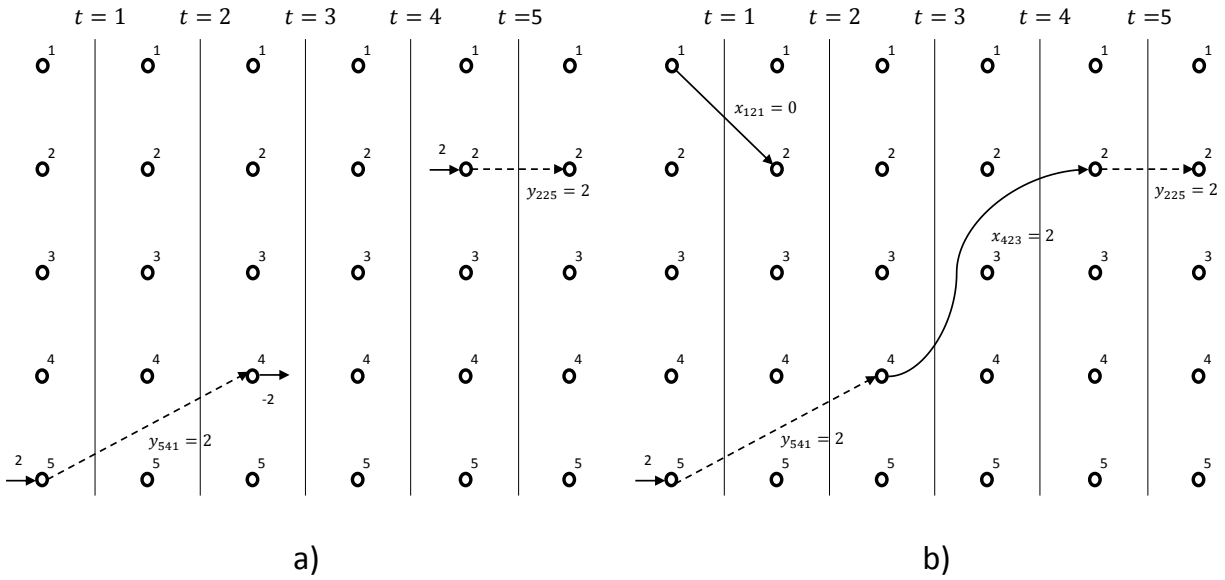


Figure 4.3: a) Second y decision. b) Third x decision.

Since the operations department does not care about vehicle's movements when they are loaded, the information they get from the sales department is when, where and how many empty vehicles are to become available (supplies) and disappear (sinks) as a consequence of loaded trips. The operations department also has information of their own fleet location and size. Figure 4.2.a illustrates the problem faced by the operations department which is a minimum cost flow problem. In trying to solve the problem, they realize that no vehicle can disappear at node $(1, 1)$ since there were none available before, and notify the sales department of the plan's lack of viability, which in turn revise their decision as in Figure 4.2.b. Finally, the operations department receives a viable plan which they can optimize by repositioning vehicles as pictured in Figure 4.3.a. The added solutions of both departments (Figure 4.3.b) is the optimal solution to the whole problem. The previous approach shows the idea of partitioning decision in Benders decomposition which we now formalize. Formulation (2.1)-(2.5) can be rewritten by setting equations (2.2) as inequality constraints $Ax \leq b$, which will be useful in cutting off half the dual space of the subproblem stemming from the Benders decomposition. Henceforth, Formulation (2.1)-(2.5) can be expressed as

$$\max \quad \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} (p_{ijv} x_{ijtv}) - \sum_{v \in V} \phi_v(x)$$

$$\text{s.t.} \quad \sum_{v \in V} x_{ijtv} \leq d_{ijt}, \qquad \forall i, j \in N, \forall t \in T$$

$$x_{ijtv} = 0, \quad if \quad A_{ijv} = 0, \quad \forall i, j \in N, \forall t \in T, \forall v \in V$$

$$x_{ijtv} \in \mathbb{Z}_+, \qquad \forall i,j \in N, \forall t \in T, \forall v \in V$$

where $\phi_v(x)$ is the objective value of the subproblem for each vehicle type $v \in V$, and is defined as follows:

$$\phi_v(x) = \min \quad \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} c_{ijv} y_{ijtv} \tag{4.2}$$

$$\text{s.t.} \quad \sum_{j \in N} (x_{ijtv} + y_{ijtv}) - \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \left(x_{ji(t-\tau_{ji})v} + y_{ji(t-\tau_{ji})v} \right) - y_{ii(t-1)v} \leq m_{itv}, \tag{4.3}$$

$$\forall i \in N, \forall t \in T \tag{4.4}$$

$$y_{ijtv} = 0, \quad if \quad A_{ijv} = 0, \qquad \forall i,j \in N, \forall t \in T \tag{4.5}$$

$$y_{ijtv} \in \mathbb{Z}_+, \qquad \forall i,j \in N, \forall t \in T \tag{4.6}$$

For a feasible $\bar{x}$, the subproblem reduces to the minimum cost flow problem (MCFP) (4.7)-(4.11) which bear the integrality property.

$$\phi_v(\bar{x}) = \min \quad \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} c_{ijv} y_{ijtv} \tag{4.7}$$

$$\text{s.t.} \quad \sum_{j \in N} y_{ijtv} - \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} y_{ji(t-\tau_{ji})v} - y_{ii(t-1)v} \leq m_{itv} - \sum_{j \in N} \bar{x}_{ijtv} + \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \bar{x}_{ji(t-\tau_{ji})v} \tag{4.8}$$

$$\forall i \in N, \forall t \in T \tag{4.9}$$

$$y_{ijtv} = 0, \quad if \quad A_{ijv} = 0, \qquad \forall i,j \in N, \forall t \in T \tag{4.10}$$

$$y_{ijtv} \in \mathbb{R}_+, \qquad \forall i,j \in N, \forall t \in T \tag{4.11}$$

The dual problem of the MCFP is obtained by means of the dual variables u associated to constraints (4.9), given by

$$\phi_v(\bar{x}) = \max \quad \sum_{i \in N} \sum_{t \in T} \left(m_{itv} - \sum_{j \in N} \bar{x}_{ijtv} + \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \bar{x}_{ji(t-\tau_{ji})v} \right) u_{it} \tag{4.12}$$

$$\text{s.t.} \quad u_{it} - u_{j(t+\tau_{ij})} \leq c_{ijv}, \qquad \forall i \in N, \forall j \in N, \forall t \in T \tag{4.13}$$

$$u_{it} \in \mathbb{R}_-, \qquad \forall i \in N, \forall t \in T \tag{4.14}$$

Note that the feasible space of the maximization problem is independent of the choice made for variables x. The feasible space for each vehicle type $v \in V$ is composed of extreme points $u_q \in Q_v$ and extreme rays $u_r \in Q_r$, where Q_v and R_v are the sets of extreme points an extreme rays, respectively, for each $v \in V$. For a given feasible solution $\bar{x}$, we have the following situations for each subproblem $v \in V$:

- The dual is unbounded: $\phi_v(\bar{x}) \to \infty$, then $\exists u_{rv} \in R_v$, such that

$$\sum_{i \in N} \sum_{t \in T} \left(m_{itv} - \sum_{j \in N} \bar{x}_{ijtv} + \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \bar{x}_{ji(t-\tau_{ji})v} \right) u_{ritv} > 0$$

  This means the primal subproblem for vehicle $v \in V$ is infeasible and in order to avoid that we impose the following cut when choosing x,

$$\sum_{i \in N} \sum_{t \in T} \left(m_{itv} - \sum_{j \in N} x_{ijtv} + \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} x_{ji(t-\tau_{ji})v} \right) u_{ritv} \leq 0$$

$$\Rightarrow \sum_{i \in N} \sum_{j \in N} \sum_{t \in T} x_{ijtv} u_{ritv} - \sum_{i \in N} \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \sum_{t \in T} x_{ji(t-\tau_{ji})v} u_{ritv} \geq \sum_{i \in N} \sum_{t \in T} m_{itv} u_{ritv}$$

  which is called a *Feasibility Cut.*

- The dual has an optimal solution: $\exists u_{qv} \in Q_v$, such that

$$\sum_{i \in N} \sum_{t \in T} \left(m_{itv} - \sum_{j \in N} \bar{x}_{ijtv} + \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \bar{x}_{ji(t-\tau_{ji})v} \right) u_{qitv}$$

  is bound on $\phi_v(x)$ which we can write:

$$\sum_{i \in N} \sum_{t \in T} \left(m_{itv} - \sum_{j \in N} x_{ijtv} + \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} x_{ji(t-\tau_{ji})v} \right) u_{qitv} \leq \phi_v$$

$$\Rightarrow \sum_{i\in N}\sum_{j\in N}\sum_{t\in T} x_{ijtv}u_{qitv} - \sum_{i\in N}\sum_{\substack{j\in N,\\ j\neq i,\\ t>\tau_{ji}}}\sum_{t\in T} x_{ji(t-\tau_{ji})v}u_{qitv} \geq \sum_{i\in N}\sum_{t\in T} m_{itv}u_{qitv} - \phi_v$$

and this constraint is called an *Optimality Cut*.

Henceforth, by considering all possible *Feasibility* and *Optimality* cuts for each vehicle $v \in V$, we obtain an equivalent formulation for problem (2.1)-(2.5) as follows

$$\max \quad \sum_{i\in N}\sum_{\substack{j\in N\\ i\neq j}}\sum_{t\in T}\sum_{v\in V}(p_{ijv}x_{ijtv}) - \sum_{v\in V}\phi_v \tag{4.15}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{ijtv} \leq d_{ijt}, \qquad \forall i,j \in N, \forall t \in T \tag{4.16}$$

$$\sum_{i\in N}\sum_{t\in T}\left(\sum_{j\in N} x_{ijtv}\bar{u}_{ritv} - \sum_{\substack{j\in N,\\ j\neq i,\\ t>\tau_{ji}}} x_{ji(t-\tau_{ji})v}\bar{u}_{ritv}\right)$$

$$\geq \sum_{i\in N}\sum_{t\in T} m_{itv}\bar{u}_{ritv}, \qquad \forall r \in R_v, \forall v \in V \tag{4.17}$$

$$\sum_{i\in N}\sum_{t\in T}\left(\sum_{j\in N} x_{ijtv}\bar{u}_{ritv} - \sum_{\substack{j\in N,\\ j\neq i,\\ t>\tau_{ji}}} x_{ji(t-\tau_{ji})v}\bar{u}_{ritv}\right)$$

$$\geq \sum_{i\in N}\sum_{t\in T} m_{itv}\bar{u}_{qitv} - \phi_v, \qquad \forall q \in Q_v, \forall v \in V \tag{4.18}$$

$$x_{ijtv} = 0, \quad if \quad , A_{ijv} = 0, \qquad \forall i,j \in N, \forall t \in T, \forall v \in V \tag{4.19}$$

$$x_{ijtv} \in \mathbb{Z}_+, \qquad \forall i,j \in N, \forall t \in T, \forall v \in V \tag{4.20}$$

$$\phi_v \geq 0, \qquad \forall v \in V \tag{4.21}$$

which is called the Benders Master Problem (MP). Given the large quantity of possible *Feasibility* and *Optimality* cuts in many practical instances, the MP can be solved by relaxing (4.17) and (4.18), and adding them iteratively. In the present work, we add the previous cuts as lazy constraints during the execution of the BC algorithm of the solver. Thus, every time the algorithm finds an incumbent integer solution $\bar{x}$, the subproblem (4.12)-(4.14) is solved with $\bar{x}$ fixed. If the subproblem is unbounded, a *Feasibility* cut is generated and added to the MP. If the subproblem is bounded and has a optimal solution, an *Optimality* cut is generated, which if violated by the current solution, is added to the MP.

4.1.1 Valid Inequalities for the VAP

In many instances of the VAP, it could be the case that the total supply of vehicles before a given time period is less than the total demand requested after that period. This creates many infeasible initial solutions for the subproblem, which could lower the efficiency of the algorithm. For instance, if we take period $t = 2$ of Figure 4.1.a, the total supply of vehicles before that period is 2 while the total aggregated demand after that period is 5. This creates an infeasible solution as described in Figure 4.1.b. To avoid these scenarios, we propose the following valid inequality to the master problem:

$$\sum_{v \in V} \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} x_{ijtv} \leq \sum_{v \in V} \sum_{i \in N} \sum_{\substack{l \in T \\ l \leq t}} m_{ilv}, \quad \forall t \in T \tag{4.22}$$

In addition, suppose that in adding the previous valid inequality for a given period t, there are highly profitable demand arcs going from $t_0 < t$ until $t_1 > t$. If carrying loads through those crossing arcs is part of the optimal solution, we want to limit the loaded movements starting after t to be limited by the total supply before t minus whatever loads are carried through those crossing arcs. Nonetheless, since we do not know which crossing arcs are part of the optimal solution, we let the solver decide which ones to use as follow:

$$\sum_{v \in V} \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} x_{ijtv} \leq \sum_{v \in V} \sum_{i \in N} \sum_{\substack{l \in T \\ l \leq t}} m_{ilv} - \sum_{v \in V} \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{\substack{l \in T \\ l < t \\ l + \tau_{ij} > t}} x_{ijlv}, \quad \forall t \in T \tag{4.23}$$

Note both inequalities can be split and added to the MP according to each vehicle type $v \in V$. However, this would significantly increase the number of constraints in the MP, which could represent a drawback when solving large-scale instances of the VAP.

4.1.2 Solving the subproblem via network flow algorithms

When implementing Benders Decomposition, it is necessary to define how to solve the subproblem which is to be solved every time an integer solution is found along the Branch-and-Bound (BB) tree. The most common way is to use general purpose solvers for solving problem (4.12)-(4.14). In some cases, when the subproblem have a special structure it is possible to get rid of general purpose solvers by using specialized algorithms. Along with valid inequalities that enrich the relaxed MP, this constitute a possible way of accelerating Benders Decomposition (Rahmaniani et al., 2017).

The problem at hand is the dual problem of a Minimum Cost Flow Problem (MCFP) whose solution can be feasible and optimal or unbounded (see Section 4.1). Since there are polynomial procedures for obtaining optimal duals from optimal flows and dual rays from primal infeasible instances, we opted to use the Network Simplex Algorithm (NSA) of the open source graph library LEMON(C++) for solving

the primal problem (i.e. MCFP) due to the following reasons: it outperformed many other algorithms in solving benchmark instances of the MCFP as stated in Kovács (2015) and its easiness of implementation and embedding within the benders decomposition algorithm.

When the solution to the dual problem is feasible and optimal, the NSA from LEMON automatically generates the optimal dual values. If this was not the case, it would have been possible to find these values by computing shortest paths from an artificial supply node (which is connected exclusively to all supply nodes through arcs with cost equal to 0) to all other nodes over the final residual graph. For a better understanding of this procedure and the concept of a residual graph see Chapter 9 of Ahuja et al. (1993).

When the dual solution is unbounded, there is no straightforward manner of querying those values from the output of the NSA, hence, we resort to checking for one sufficient condition and/or solving a special case of the MCFP (namely, the Maximum Flow Problem) in order to compute the unbounded rays. When solving the MFP, we used the push-relabel algorithm implementation of LEMON.

First, it is useful to understand when an instance for formulation (4.7)-(4.11) is infeasible. Figure 4.4.a shows an instance of two nodes of the MCFP, where b_i units of flow (empty vehicles) are supplied through node i and b_k units of flow (empty vehicles) are demanded on node k. This instance is infeasible as there is not enough flow arriving to node k. A dual ray for this problem equals $u_i = -1, u_k = -1$. Figure 4.4.b shows an instance of three nodes whose total supply ($b_i + b_j$) is greater than the total demand (b_k), yet is infeasible as there is not enough flow arriving to node k. A dual ray for this problem equals $u_i = -1, u_k = -1, u_j = 0$.

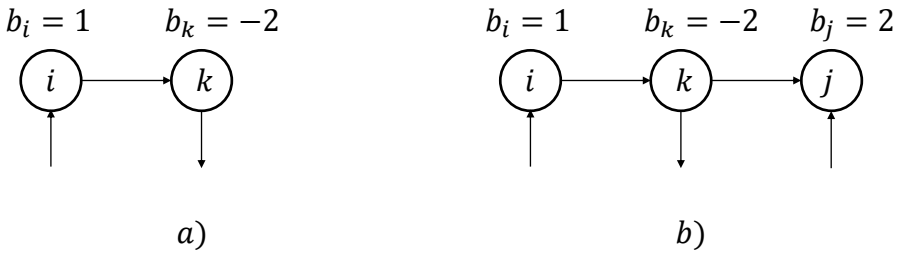


Figure 4.4: a) Infeasible instance of the MCFP with two nodes. b) Infeasible instance of the MCFP with three nodes.

In topologically sorted graphs like those in Figures 4.4.a and 4.4.b, it suffices to find a node k such that $b_k < 0$ and $\sum_{l<k} b_l < |b_k|$, to prove the instance is infeasible and build a dual ray. Therefore, the dual ray of infinite improvement, when such node k has been found, is $u_0 = .. = u_k = -1$ and $u_{k+1} = ... = 0$. In a graph with N nodes, this ray is of infinite improvement as

$$\sum_{l \in N} u_l b_l = \sum_{l \leq k} u_l b_l + \sum_{l > k} u_l b_l > 0 \tag{4.24}$$

given $\sum_{l \leq k} u_l b_l > 0$ and $\sum_{l > k} u_l b_l = 0$. In addition, since it is a ray it should lie in the recession cone $A^T u \leq 0$ (where A^T denotes the transpose of the incidence matrix A), for which we check the sign of the difference between the duals of any two nodes when: both are in the set $\{0, .., K\}$ (equation (4.25)), both are in the set $\{k+1, .., N\}$ (equation (4.26)) and one node belongs to $\{0, .., K\}$ while the other to $\{k+1, .., N\}$ (equation (4.27)).

$$\forall i, j \in \{0, .., k\} \qquad u_i - u_j \leq 0 \tag{4.25}$$

$$\forall i, j \in \{k+1, .., N\} \qquad u_i - u_j \leq 0 \tag{4.26}$$

$$\forall i \in \{0, .., k\}, j \in \{k+1, .., N\} \qquad u_i - u_j \leq 0 \tag{4.27}$$

Checking for this condition has an asymptotic cost of $\mathcal{O}(N)$. Unfortunately, the previous condition is only sufficient as it may not hold true for a given topological order of the same graph. For instance, Figure 4.5 shows two different topological orders of a infeasible instance wherein the condition holds true for the top graph while it does not for the bottom graph.

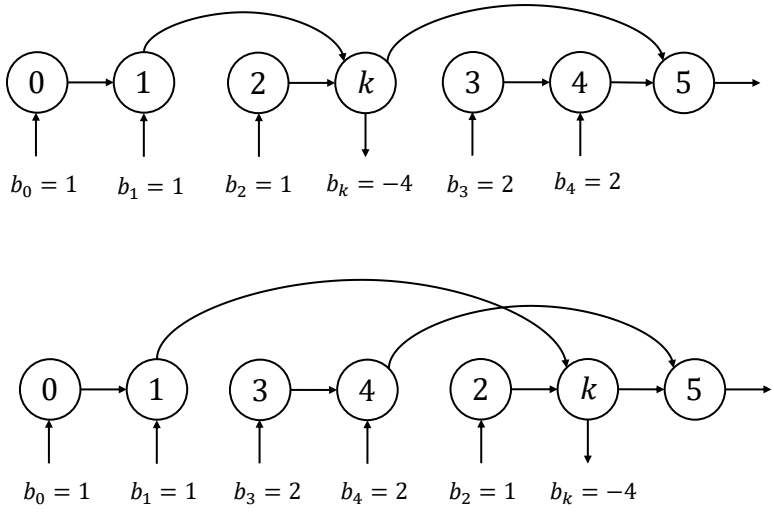


Figure 4.5: Two topological order of the same infeasible instance of the MCFP.

Situations like the one in Figure 4.5 can be overcome by backtracking (bold arrows and circles) on a given (set of) demand node(s) and checking if the total supply in the inspected nodes is less than the total demand. For instance, Figure 4.5 shows how backtracking starting from k proves the instance

to be infeasible. By using a search graph algorithm on the reversed graph (same graph with reversed arcs) departing from k, we discover that the only nodes being able to supply k are $\{0, 1, 2\}$ whose total supply (3) is not enough to cover the demand of k (4). Therefore, we can find an unbounded ray of infinite improvement by setting $u_k = -1, \forall k \in S$ and $u_k = 0, \forall l \in \bar{S}$, where S is a set of demand nodes plus all nodes inspected when backtracking from these demand nodes and $\bar{S}$ is the complement of S. If $\sum_{l \in S} b_l < 0$, then this ray is of infinite improvement as

$$\sum_{l \in N} u_l b_l = \sum_{l \in S} u_l b_l + \sum_{l \in \bar{S}} u_l b_l > 0 \tag{4.28}$$

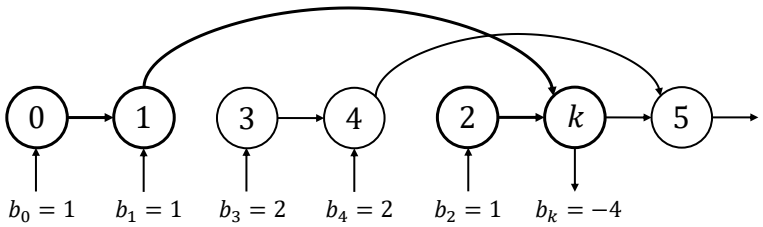


Figure 4.6: Infeasible instance showing how backtracking from a demand node can be useful in identifying infeasibility.

It also lies in the recession cone $A^T u \leq 0$ as

$$\forall i, j \in S \qquad u_i - u_j \leq 0 \tag{4.29}$$

$$\forall i, j \in \bar{S} \qquad u_i - u_j \leq 0 \tag{4.30}$$

$$\forall i \in S, j \in \bar{S} \quad u_i - u_j \leq 0 \tag{4.31}$$

Note that the pair (i, j) where $i \in \bar{S}, j \in S$ is not possible due to the fact that an existing arc between i and j would have enabled i to be inspected in the backtracking. Finding single demand nodes and backtracking to evaluate if there is enough supply to that demand node is not sufficient to prove that an instance is infeasible. Figure 4.7 shows an instance of a graph with two demand nodes and we can observe that by backtracking from these nodes individually the above mentioned condition for infeasibility does not hold. By contrast, Figure 4.8 shows that jointly backtracking from both demand nodes attest infeasibility as the total demand (not individual demands) cannot be supplied from the reachable supply nodes. Therefore, to prove infeasibility it is necessary to find the correct set of demand nodes that cannot be serviced from all their reachable supply nodes.

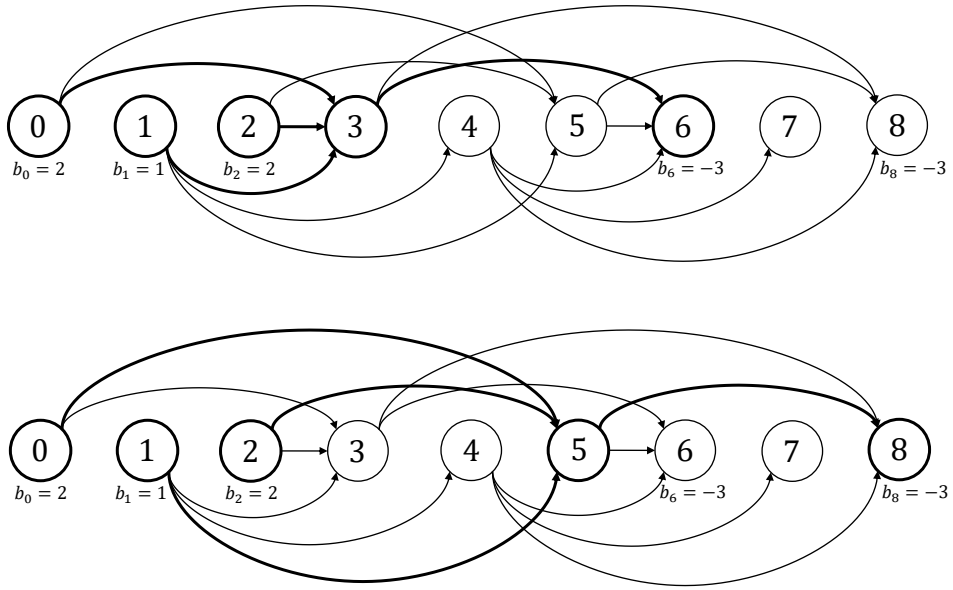


Figure 4.7: Instance of a graph where backtracking on individual demand nodes may not prove infeasibility.

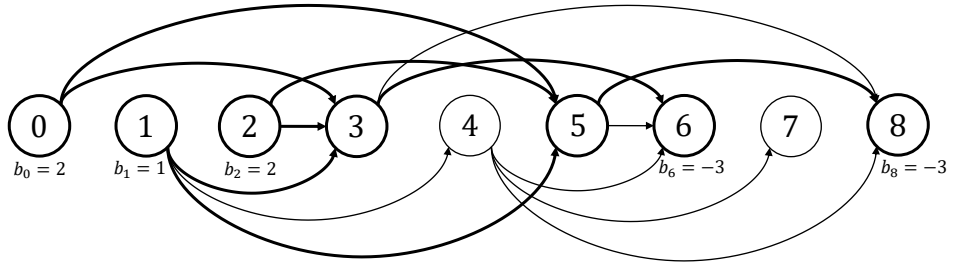


Figure 4.8: Instance of a graph where infeasibility is proved by analyzing sets of demand nodes.

Even though there are $2^{|D|}$ subsets of demand nodes, where D is the total number of demand nodes ($b < 0$) in a graph, and brute-force searching could be computationally expensive, we can solve a simpler and polynomially solvable problem in order to find the correct subset of demand nodes that enables us to come up with a unbounded ray, namely, the Maximum Flow Problem (MFP). Figure 4.9 shows a modified directed acyclic graph where all demand (supply) nodes are connected to an artificial demand (supply) node and the artificial arcs connecting the demand (supply) nodes to the artificial demand (supply) have capacities w equal to $|b|$. All other arcs are uncapcitated as is the VAP case.

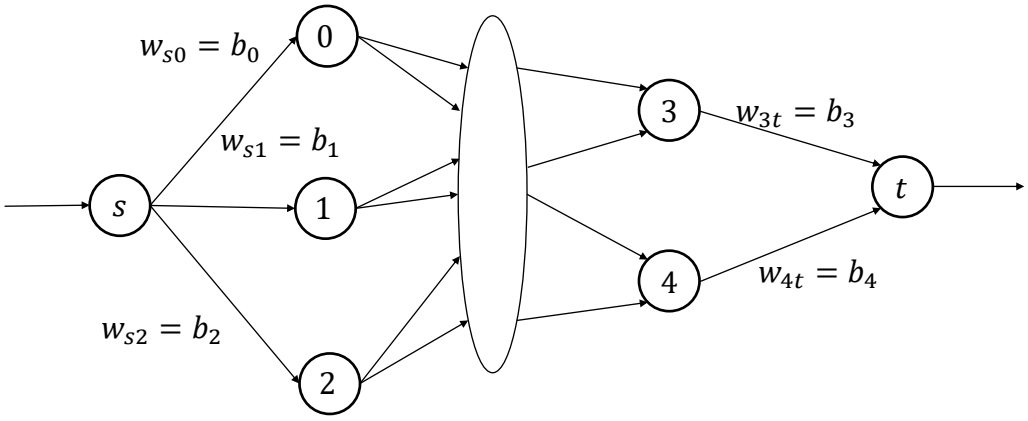


Figure 4.9: Modified graph with one supply and demand node in order to solve the Maximum Flow Problem.

An optimal solution to the MFP, that is also a feasible solution to the MCFP, consists of a vector flow where all artificial demand arcs are saturated, i.e. there is no unused capacities among these arcs. On the other hand, if the MCFP is infeasible, the optimal solution to the MFP will reveal a subset of artificial demand arcs with unused capacity. Figure 4.10 shows the optimal solution to a MFP where A denotes the saturated artificial supply arcs, B denotes the unsaturated artificial supply arcs, C denotes the unsaturated artificial demand arcs and D denotes the unsaturated artificial supply arcs. The sets S and $\bar{S}$ represent the partition of the graph's nodes whose crossing arcs constitute the set of arcs with minimum total capacity, i.e. the minimum cut.

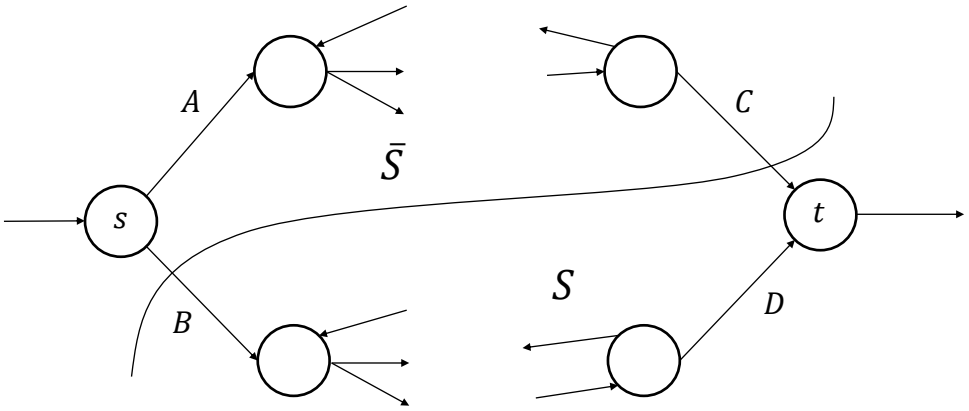


Figure 4.10: General scheme of an optimal solution for the MFP.

If we backtrack with a searching algorithm departing from all nodes in D, we will end up with the set of nodes S as the one depicted in Figure 4.10. Backtracking from the nodes in D can not end up in any node in $\bar{S}$ as nodes in A have unused capacity that could have been used in filling up nodes in D, hence contradicting the fact of that being a maximum flow. Therefore, the set S contains the set of unfulfilled

demand nodes and all its possible suppliers which was the condition we were initially looking for in order to build an unbounded ray. Again, we can set the unbounded ray of infinite improvement by setting $u_k = -1, \forall k \in S$ and $u_k = 0, \forall l \in \bar{S}$. This is a ray of infinite improvement as

$$\sum_{l \in N} u_l b_l = \sum_{l \in S} b_l = \sum_{l \in G_+ \cap S} b_l - \sum_{l \in G_- \cap S} |b_l| > \sum_{l \in G_+ \cap S} b_l - \sum_{l \in G_- \cap S} f_{it} = 0$$

where G_+,G_- and f_{it} are the set of all supply nodes, all demand nodes and the flow at optimality (in the context of MFP) of arcs going from i to t, respectively. It also lies in the recession cone $A^T u \leq 0$ as per equations (4.29)-(4.31). Knowing the asymptotic cost of the MFP algorithm, the preflow push-relabel to be more specific, is $\mathcal{O}(N^2\sqrt{E})$ where N and E are the number of nodes and edges of the graph, we decided to organize the search for the unbounded ray, once the instance has proven to be infeasible by the NSA, as presented in Algorithm 1.

Algorithm 1 Unbounded Ray

```
net-supply = 0
for l ← 0, N do
    net-supply += b_l
    if net-supply < 0 then
        Build Ray
        EXIT
    end if
end for
Solve Maximum Flow Problem
Build Ray
EXIT
```

4.2 Computational Results

To evaluate the computational performance of Benders Decomposition as applied to the VAP, we test two ways of applying Benders Decomposition and compare their performance to the standalone BC of the optimization solver IBM CPLEX Optimization Studio 12.8.1. in solving the compact formulation. The most straighforward way of implementing Benders Decomposition is through the automatic Benders strategy of CPLEX in which annotations on variables are made over the compact formulation definition. We use the following user defined annotations: x variables were given the 0 annotation (annotations with 0 values assign the variables to the MP) and y variables for vehicle type $v \in V$ were given the $v + 1$ annotation (annotations with values greater than 0 assign the variables to its corresponding subproblem). Then, CPLEX manage the interaction between MP and the subproblem as well as solving the subproblem. CPLEX documentation claims this strategy can be helpful in some problems as certain types of subproblem can be solved in parallel. Another way of implementing Benders De-

composition is through callback functions in which the user has to implement the separation problem, in this case the subproblem resulting from Benders decomposition, and add Benders cuts whenever an infeasible integer solution is found along the BC tree. In the second case, we used the Network Simplex Algorithm from LEMON graph library, for solving the separation problem as explained in Section 4.1.2. All methods were implemented in C++ using the Concert library and LEMON graph library. All experiments reported in this section were run in a PC with CPU Intel®Core i7-4790S 3.20GHz and 16 GB of RAM. In the experiments, we solve small-scale randomly generated instances to test the viability of this method. In naming each instance we use the following notation: *terminals-periods-vehicles-requests*. The instances were created in the following way using a uniformly distributed random number generator:

- We created a square with dimensions equal to the number of periods $|T|$.
- For each terminal, coordinates were randomly generated within this square .
- Euclidian distances were calculated for each pair of terminals, and they were truncated to its lowest integer value to obtain the integer travel times τ_{ijv}.
- Costs for empty movements were calculated by multiplying the travel times τ_{ijv} by a random number in the range $[0, 20]$.
- Profits for loaded movements were calculated by adding a random number in the range $[1, 20]$ to the cost of empty movements calculated in the previous step.
- Demands were randomly generated in the range $[1, 10]$. The departure and destination terminals, and starting period were randomly generated according to the number of terminals and periods, respectively.
- The entry of each vehicle m_{itv} was randomly generated according to the number of terminals and periods, respectively. None of these instances has $m_{itv} = -1$.
- Restriction of movements were generated randomly by picking a random number p between 1 and $(N * N)/2$. Then, choose randomly p arcs out of $N * N$ arcs and set its value $A_{ijv} = 0$.

Tables 4.1 and 4.2 shows the computational times of the proposed method. Table 4.1 contains results for instances with the number of terminals ranging from 10 to 19, number of vehicles from 20 to 50, 10 time periods and 20 loads. Table 4.2 contains results for instances with the number of terminals ranging from 20 to 29, number of vehicles from 100 to 150, 20 time periods and 200 loads. Finally, Table 4.3 shows the percentage of computational time spent on solving the subproblem and the number of Benders cuts generated by the proposed BBC for instances of Table 4.1. A time limit of 3600 seconds

was imposed on each run. A sign "*" in the tables indicates that CPLEX could not solve or even mount the model due to lack of computer memory. Columns in these tables refer to:

- Instance is the name of the instance (number of terminals - number of periods - number of vehicles - number of requests).
- IP is the optimal ILP solution value of VAP.
- CPU IP is the time taken by the standalone BC of CPLEX to solve each ILP instance to optimality.
- CPU Be is the time taken by CPLEX with automatic Benders.
- CPU BeV1 is the time taken by CPLEX with automatic Benders and valid inequalities (4.22).
- CPU BeV2 is the time taken by CPLEX with automatic Benders and valid inequalities (4.23).
- CPU LEM is the time taken by the proposed BBC.
- CPU LEMV1 is the time taken by the proposed BBC and valid inequalities (4.22).
- CPU LEMV2 is the time taken by the proposed BBC and valid inequalities (4.23).
- % LEM is percentage of computational time spent on solving the subproblem by the proposed BBC.
- F LEM is the number of feasibility cuts generated by the proposed BBC.
- O LEM is the number of optimality cuts generated by the proposed BBC.
- % LEMV1 is percentage of computational time spent on solving the subproblem by the proposed BBC and valid inequalities (4.22).
- F LEMV1 is the number of feasibility cuts generated by the proposed BBC and valid inequalities (4.22).
- O LEMV1 is the number of optimality cuts generated by the proposed BBC and valid inequalities (4.22).
- % LEMV2 is percentage of computational time spent on solving the subproblem by the proposed BBC and valid inequalities (4.23).
- F LEMV2 is the number of feasibility cuts generated by the proposed BBC and valid inequalities (4.23).

Table 4.1: Results of the small-scale instances with terminal in the range [10,19] for different implementations of the Benders Decomposition as applied to the VAP.

Instance	IP	CPU IP	CPU Be	CPU BeV1	CPU BeV2	CPU LEM	CPU LEMV1	CPU LEMV2
10-10-20-20	1937	0.52	0.152	0.145	0.163	6.24	3.58	7.79
11-10-20-20	1584	0.50	0.135	0.134	0.141	9.73	2.85	13.71
12-10-20-20	904	0.59	0.181	0.166	0.208	37.36	23.07	18.92
13-10-20-20	1259	0.18	0.251	0.235	0.254	19.57	25	27.94
14-10-20-20	633	0.21	0.178	0.173	0.194	2.37	1.91	2.89
15-10-20-20	560	0.26	0.253	0.255	0.251	22.55	9.35	22.26
16-10-20-20	736	0.46	0.319	0.289	0.314	72.11	221.59	44.99
17-10-20-20	632	0.30	0.295	0.291	0.289	8.83	30.12	61.71
18-10-20-20	1143	0.29	0.338	0.306	0.355	42.31	24.17	31.39
19-10-20-20	905	0.33	0.312	0.304	0.314	12.18	8.86	13.33
10-10-20-25	1857	0.12	0.136	0.123	0.120	6.49	1.59	2.11
11-10-20-25	659	0.13	0.138	0.130	0.142	2.97	2.63	4.45
12-10-20-25	396	0.21	0.150	0.148	0.155	1.45	1.36	1.58
13-10-20-25	669	0.26	0.207	0.214	0.216	84.66	136.16	152.34
14-10-20-25	1292	0.36	0.280	0.268	0.302	74.49	97.86	184.51
15-10-20-25	1202	0.27	0.287	0.261	0.257	255.93	257.64	475.13
16-10-20-25	1479	0.29	0.285	0.288	0.308	60.86	42.15	98.21
17-10-20-25	1611	0.31	0.384	0.311	0.292	483.79	305.4	352.26
18-10-20-25	291	0.41	0.271	0.279	0.293	3.56	3.45	3.95
19-10-20-25	535	0.36	0.380	0.347	0.390	19.67	13.28	25.63
10-10-20-30	501	0.10	0.096	0.094	0.100	1.47	1.56	2.2
11-10-20-30	1596	0.16	0.174	0.148	0.161	13.31	20.86	17.37
12-10-20-30	335	0.17	0.164	0.208	0.181	3.2	3.29	4.22
13-10-20-30	580	0.17	0.167	0.162	0.168	62.77	76.82	98.1
14-10-20-30	574	0.31	0.230	0.214	0.256	2.96	2.88	4.04
15-10-20-30	1451	0.26	0.279	0.261	0.269	18.87	13.89	29.3
16-10-20-30	1788	0.28	0.320	0.350	0.324	11.42	10.52	10.72
17-10-20-30	424	0.28	0.285	0.299	0.311	7.81	7.1	8.25
18-10-20-30	1835	0.29	0.454	0.369	0.365	113.27	602.71	458.71
19-10-20-30	1402	0.45	0.439	0.439	0.393	32.73	25.2	34.86
10-10-20-35	2464	0.16	0.142	0.308	0.143	12.12	12.84	15.71
11-10-20-35	2238	0.29	0.158	0.157	0.143	8.92	11.9	12.97
12-10-20-35	669	0.18	0.164	0.163	0.176	2	2.21	3.42
13-10-20-35	266	0.16	0.167	0.193	0.177	1.92	1.81	1.99
14-10-20-35	2057	0.21	0.245	0.219	0.221	11.07	13.66	15.61
15-10-20-35	1679	0.23	0.197	0.219	0.234	4.08	5.02	5.84
16-10-20-35	2037	0.27	0.275	0.262	0.300	8	4.48	6.3
17-10-20-35	2347	0.32	0.371	0.347	0.361	131.14	192.71	450.31
18-10-20-35	991	0.78	0.354	0.370	0.401	51.77	75.01	61.76
19-10-20-35	691	0.38	0.405	0.404	0.421	1074.71	3071.66	2147.35
10-10-20-50	421	0.15	0.147	0.157	0.195	2.09	1.73	2.16
11-10-20-50	2701	0.26	0.252	0.281	0.311	177.49	87.82	70.09
12-10-20-50	811	0.16	0.160	0.181	0.164	12.81	12.49	13.52
13-10-20-50	678	0.33	0.191	0.167	0.201	164.81	310.65	246.88
14-10-20-50	355	0.65	0.215	0.235	0.244	2.02	2.1	2.38
15-10-20-50	623	0.36	0.420	0.363	0.440	677.55	578.2	1380.42
16-10-20-50	1562	0.33	0.342	0.377	0.388	*	*	*
17-10-20-50	2202	0.37	0.373	0.418	0.422	197.44	148.09	149.91
18-10-20-50	550	0.35	0.485	0.509	0.568	97.64	126.16	143.28
19-10-20-50	2591	0.61	0.480	0.450	0.511	131.61	312.06	369.34

Table 4.2: Results of the small-scale instances with terminal in the range [20,29] for testing the Benders Decomposition as applied to the VAP.

Instance	IP	CPU IP	CPU Be	CPU BeV1	CPU BeV2
20-20-100-200	14471	16.23	24.073	26.900	26.690
20-20-130-200	31327	19.13	51.290	48.275	44.544
20-20-150-200	34618	22.10	58.536	60.649	61.194
21-20-100-200	2585	19.09	17.142	17.557	18.091
21-20-130-200	4056	21.65	39.251	38.586	42.458
21-20-150-200	27322	23.34	44.581	45.440	44.099
22-20-100-200	4902	23.76	39.907	38.630	40.453
22-20-130-200	12301	25.56	52.665	53.395	61.025
22-20-150-200	10971	34.33	71.556	68.402	65.962
23-20-100-200	14132	19.48	21.500	21.373	23.252
23-20-130-200	21745	27.30	51.433	51.442	52.676
23-20-150-200	23358	31.67	55.262	55.943	56.120
24-20-100-200	10155	24.04	34.142	36.996	36.251
24-20-130-200	3486	42.33	35.293	31.914	41.655
24-20-150-200	18156	52.91	119.447	105.599	110.724
25-20-100-200	9335	23.05	30.674	27.766	31.509
25-20-130-200	33905	38.92	83.977	67.335	79.588
25-20-150-200	32508	37.72	64.953	58.269	65.790
26-20-100-200	9815	22.13	47.205	42.733	43.493
26-20-130-200	16892	33.17	70.801	63.701	66.035
26-20-150-200	3731	67.78	52.401	50.378	57.074
27-20-100-200	21090	32.24	53.283	51.636	53.628
27-20-130-200	8456	55.14	133.089	125.644	138.139
27-20-150-200	26704	44.40	76.242	81.807	83.249
28-20-100-200	3641	42.01	50.944	54.572	125.903
28-20-130-200	25355	36.32	64.809	61.111	57.876
28-20-150-200	11222	57.47	125.888	126.390	207.826
29-20-100-200	5994	31.58	48.046	44.500	44.376
29-20-130-200	8978	49.86	142.506	121.911	232.190
29-20-150-200	7977	44.10	66.834	68.652	70.362

- O LEMV2 is the number of optimality cuts generated by the proposed BBC and valid inequalities (4.23).

From Tables 4.1-4.2 we observe that 34, 34 and 28 out of 80 instances are solved faster with automatic Benders strategy, automatic Benders strategy with valid inequalities (4.22) and automatic Benders strategy with valid inequalities (4.23), respectively, than with the compact formulation. On the other hand, the proposed BBC was not able to solve any instance faster than the automatic Benders strategy or the compact formulation. In particular, instance 16-10-20-50 and all instances with terminals in the range [20,29] could not be solved to optimality with the proposed BBC due to CPLEX running out of memory. In comparing the 3 ways of solving the problem with the automatic Benders strategy, we observe that 49 and 22 out of 80 instances were solved faster with valid inequalities (4.22) and (4.23), respectively, than without any valid inequality. Regarding the proposed BBC, 26 and 9 out of 49 instances were solved faster with valid inequalities (4.22) and (4.23), respectively, than without any valid inequality. This suggests that valid inequality (4.22) may help enriching the Benders MP in order to obtain more efficient computational results, whereas, we suspect valid inequality (4.23) makes the MP harder to solve, hence, the results are worse in both Benders approaches for this valid inequality.

From Table 4.3, we observe that the maximum percentage of computational time spent by the subproblem with the proposed BBC without valid inequalities, with valid inequalities (4.22) and with valid inequalities (4.23) are 32.36%, 31.51%, and 33.85%, respectively. Furthermore, the average percentage of computational time spent by the subproblem in all three cases are 9.24%, 8.56% and 7.26%, respectively. This shows that the majority of time is spent in solving and processing the MP along the BC tree. Regarding the generated feasibility cuts, we observe that in all instances there was a reduction in the number of feasibility cuts when valid inequalities (4.22) and (4.23) were enforced at the root. Yet, as mentioned earlier, the computational times were not better in the case of valid inequality (4.23). Regarding the generated optimality cuts, less than half the instances showed an increase in the number of optimality cuts when comparing the BBC with (4.22) and (4.23) to the BBC without valid inequalities. As we will see in the next chapter, CPLEX in solving the compact formulation is able to solve instances larger than the ones in Table 4.2. By contrast, Benders with both approaches fell short in memory when trying to solve instances larger than those in Table 4.2. The previous results suggest we need to research better ways to enrich the MP in order to avoid generating a large number of cuts and improve the convergence of the proposed method.

4.3 Final considerations and next steps

In this chapter we have presented results from applying Benders Decomposition to the VAP. We have presented the formal decomposition and found that the subproblem is a multiple origin-destination

Table 4.3: Subproblem times and number of Benders cuts of the small-scale instances with terminal in the range [10,19] for different implementations of the Benders Decomposition as applied to the VAP.

Instance	% LEM	F LEM	O LEM	% LEMV1	F LEMV1	O LEMV1	% LEMV2	F LEMV2	O LEMV2
10-10-20-20	4.17%	337	63	7.82%	306	61	3.08%	276	70
11-10-20-20	30.63%	353	809	9.82%	321	43	28.15%	288	937
12-10-20-20	1.31%	343	69	1.73%	288	59	1.90%	249	69
13-10-20-20	2.45%	350	76	1.80%	335	72	1.79%	282	72
14-10-20-20	18.99%	327	27	24.08%	268	55	12.11%	244	31
15-10-20-20	26.34%	350	821	5.88%	296	45	22.51%	251	677
16-10-20-20	1.10%	376	91	0.31%	381	103	1.64%	276	77
17-10-20-20	7.93%	375	49	31.51%	340	1056	33.85%	276	2283
18-10-20-20	1.91%	336	44	2.40%	283	37	2.33%	239	46
19-10-20-20	8.37%	370	42	10.38%	330	41	6.08%	251	48
10-10-20-25	32.36%	364	622	14.47%	273	29	10.90%	264	43
11-10-20-25	13.80%	430	46	12.55%	386	47	6.29%	294	44
12-10-20-25	24.83%	390	0	22.79%	372	0	19.62%	267	0
13-10-20-25	11.12%	498	1755	0.37%	381	106	5.44%	325	1545
14-10-20-25	0.97%	488	74	0.69%	431	109	0.30%	363	86
15-10-20-25	0.34%	477	85	0.31%	426	92	0.16%	338	86
16-10-20-25	1.58%	408	116	1.80%	346	76	0.77%	341	82
17-10-20-25	0.19%	454	67	0.29%	429	77	0.23%	367	62
18-10-20-25	17.13%	360	0	20.00%	284	0	15.44%	218	0
19-10-20-25	11.34%	430	193	7.15%	372	79	4.49%	345	90
10-10-20-30	17.69%	453	39	16.67%	337	42	10.00%	262	39
11-10-20-30	3.46%	505	77	2.40%	392	75	2.25%	316	69
12-10-20-30	14.06%	607	0	19.15%	523	0	14.69%	414	0
13-10-20-30	2.47%	506	356	0.92%	419	101	0.51%	329	70
14-10-20-30	21.96%	538	50	24.31%	487	45	16.09%	330	35
15-10-20-30	4.35%	543	62	4.75%	421	58	2.59%	384	52
16-10-20-30	7.62%	556	69	8.65%	484	56	7.00%	365	52
17-10-20-30	11.01%	476	60	12.54%	410	58	9.94%	299	57
18-10-20-30	1.17%	489	97	0.16%	384	72	0.20%	321	72
19-10-20-30	3.70%	541	55	4.40%	422	61	3.04%	361	66
10-10-20-35	2.48%	478	63	3.19%	438	58	2.23%	329	58
11-10-20-35	5.04%	577	64	3.53%	420	54	2.70%	378	47
12-10-20-35	19.00%	445	41	18.10%	340	42	11.99%	278	42
13-10-20-35	22.92%	460	0	21.55%	363	0	16.08%	258	0
14-10-20-35	6.78%	610	65	4.25%	420	52	3.52%	333	52
15-10-20-35	19.36%	512	35	15.94%	391	48	12.16%	313	45
16-10-20-35	13.00%	665	72	21.43%	509	69	12.86%	364	57
17-10-20-35	0.91%	648	94	0.58%	537	104	0.22%	449	84
18-10-20-35	2.14%	543	98	1.61%	518	81	2.33%	407	89
19-10-20-35	0.15%	615	112	0.05%	509	143	0.06%	404	101
10-10-20-50	15.79%	714	0	19.65%	487	0	14.81%	388	0
11-10-20-50	0.25%	750	82	0.61%	652	83	0.80%	506	83
12-10-20-50	4.45%	719	42	4.48%	530	42	3.40%	379	48
13-10-20-50	0.53%	794	74	0.22%	538	57	0.31%	460	109
14-10-20-50	31.68%	627	0	31.43%	479	0	19.75%	337	0
15-10-20-50	0.17%	971	99	0.20%	748	112	0.08%	630	104
16-10-20-50	*	960	117	*	679	125	*	549	100
17-10-20-50	0.77%	798	96	1.01%	638	104	0.81%	485	98
18-10-20-50	1.44%	893	65	1.16%	569	92	0.89%	479	88
19-10-20-50	1.32%	795	94	0.56%	602	102	7.32%	512	2365

minimum cost flow problem. This problem bears the integrality property, which allows us to apply the classic Benders decomposition. We implemented this approach with the embedded Benders strategy of

CPLEX and Benders using lazy constraints. In the case of Benders using lazy constraints, we resort to Network Flow algorithms to speed up the process of generating constraints. However, both Benders approaches are insufficient in processing and solving large-scale instances as CPLEX ran out of memory in solving instances with more than 30 terminals.

Chapter 5

A Branch-and-Price algorithm for the VAP based on the arc-demand formulation

One of the objectives of this work is to propose an exact solution method for solving large-scale realistic instances of the VAP. Given the promising results obtained in Cruz (2017) in applying the Primal-Dual Column Generation Method (PDCGM) from Gondzio et al. (2016) for solving the VAP, this work is extended to a BP method for obtaining the integer optimal solution.

5.1 Dantzig-Wolfe Decomposition

In this section, the Dantzig-Wolfe (DW) decomposition and the column generation (CG) method are presented for the VAP based on formulation (2.1)-(2.5). We chose to decompose according to types of vehicles as it significantly reduces the ILP model. This reformulation has $|N| \times |N| \times |T| + |V|$ constraints compared to $|N| \times |T| \times |V| + |N| \times |N| \times |T| + |V|$ constraints from the compact formulation of Section 2.3. Furthermore, even though the number of variables of the reformulation may be larger than that of the compact formulation, in practice that number is smaller as only columns (hence variables) with positive reduced cost are added to the model in a columns generation approach. From a practical point of view, it is best for operations control to create programs according to vehicle groups or individual vehicles, and from a modelling perspective, quality results have been obtained in related network problems when decomposing programs according to vehicle characteristics (Cruz et al., 2019; Cruz, 2017; Munari et al., 2019). In this way, consider the linear relaxation of problem (2.1)-(2.5) and leave the demand-satisfying constraints (2.3) as the coupling constraints. Then, the remaining constraints can be

grouped into the sets of solutions $X_v, \forall v \in V$, given by:

$$X_v = \left\{ \begin{array}{lll} (x_v, y_v) | & \displaystyle\sum_{j \in N} (x_{ijtv} + y_{ijtv}) - \sum_{\substack{k \in N, \\ k \neq i, \\ t > \tau_{ki}}} \left(x_{ki(t-\tau_{ki})v} + y_{ki(t-\tau_{ki})v} \right) & \\ & - y_{ii(t-1)v} = m_{itv}, & \forall i \in N, \forall t \in T, \\ & x_{ijtv} = 0 \wedge y_{ijtv} = 0, \text{ if } A_{ijv} = 0, & \forall i, j \in N, \forall t \in T, \\ & x_{ijtv} \in \mathbb{R}^+, y_{ijtv} \in \mathbb{R}^+, & \forall i, j \in N, \forall t \in T. \end{array} \right\}$$

Thus, the resulting equivalent formulation to (2.1)–(2.5) writes as:

$$\begin{array}{lll} \max & \displaystyle\sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} (p_{ijv} x_{ijtv} - c_{ijv} y_{ijtv}) & \\ \text{s.t.:} & \displaystyle\sum_{v \in V} x_{ijtv} \leq d_{ijt}, & \forall i, j \in N, \forall t \in T \\ & (x_v, y_v) \in X_v, & \forall v \in V. \end{array}$$

By the representation theorem (see e.g. Bertsimas and Tsitsiklis (1997)), any solution $(x_v, y_v) \in X_v$ can be described as a linear convex combination of extreme points and a linear combination of extreme rays of X_v. Note that the set X_v is described by flow conservation equations where the left-hand side is the incidence matrix of a network, which in turn defines trees over a DAG for a feasible solution; the right-hand side imposes a limit on the amount of vehicles flowing out of node (i, t). Therefore, the sets X_v are bounded and the extreme rays are omitted in the description of the solution:

$$(x_v, y_v) = \sum_{q \in Q_v} (\bar{x}_{vq}, \bar{y}_{vq}) \lambda_{vq},$$
$$\sum_{q \in Q_v} \lambda_{vq} = 1, \lambda_{vq} \geq 0,$$

where $(\bar{x}_{vq}, \bar{y}_{vq})$ denotes the extreme points of X_v, and Q_v is the set of all extreme points. When substituting this representation in the linear relaxation of problem above, the result is the Master Problem (MP):

$$\max \quad \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} \left(p_{ijv} \sum_{q \in Q_v} \bar{x}_{qijtv} \lambda_{vq} - c_{ijv} \sum_{q \in Q_v} \bar{y}_{qijtv} \lambda_{vq} \right) \tag{5.1}$$

$$\text{s.t.:} \quad \sum_{v \in V} \sum_{q \in Q_v} \bar{x}_{qijtv} \lambda_{vq} \leq d_{ijt}, \qquad \forall i, j \in N, \forall t \in T, \qquad (u_{ijt}) \tag{5.2}$$

$$\sum_{q \in Q_v} \lambda_{vq} = 1, \qquad \forall v \in V, \qquad (w_v) \tag{5.3}$$

$$\lambda_{vq} \geq 0, \qquad \forall v \in V, \forall q \in Q_v, \tag{5.4}$$

The reduced cost of each variable λ_{vq} is $((p_v \bar{x}_{vq} - c_v \bar{y}_{vq}) - u\bar{x}_{vq}) - w_v$, where u and w_v represent the vector of dual variables of the coupling and convexity constraints, respectively. Given the huge size of Q_v and the fact that not all variables are part of the optimal basic feasible solution, variables are iteratively added by finding the columns with a positive reduced cost considering the following pricing subproblem $Z_{sp(v)}$:

$$Z_{sp(v)} = \max \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} ((p_{ijv} x_{ijtv} - c_{ijv} y_{ijtv}) - u_{ijt} x_{ijtv}) \tag{5.5}$$

$$\text{s.t.:} \quad \sum_{j \in N} (x_{ijtv} + y_{ijtv}) - \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \left(x_{ji(t-\tau_{ji})v} + y_{ji(t-\tau_{ji})v} \right) \tag{5.6}$$

$$- y_{ii(t-1)v} = m_{itv}, \qquad \forall i$$

$$x_{ijtv} = 0 \wedge y_{ijtv} = 0, \text{ if } A_{ijv} = 0, \qquad \forall i, j \in N, \forall t \in T, \tag{5.7}$$

$$x_{ijtv} \in \mathbb{R}_+, y_{ijtv} \in \mathbb{R}_+, \qquad \forall i, j \in N, \forall t \in T, \tag{5.8}$$

where the reduced cost of variable λ_{vq} is $Z_{sp(v)} - w_v$. This allows us to initialize the MP with just a subset of columns, resulting in what is called the Restricted Master Problem (RMP), and iteratively generate new columns from the dual solutions of the RMP. This procedure is known as the Column Generation method, which converges to an optimal solution of the MP. Note that the solution obtained through the CG corresponds to an optimal solution of the linear programming (LP) relaxation of the original problem (2.1)–(2.5), in this particular case.

5.2 Primal-Dual Column Generation Method

The standard CG based on optimal dual solutions presents several drawbacks, specially when the simplex method is used to solve the RMP (Lübbecke and Desrosiers, 2005; Vanderbeck, 2005). Among them are: slow convergence near the optimal solution (tailing-off effect); the first iterations produce irrelevant columns and dual bounds due to poor dual information at the onset (heading-in effect); degeneracy in the primal and hence multiple optimal solutions in the dual: the value of the RMP remains constant for several iterations (plateau effect); instability in the dual solutions that jump from one extreme value to another (yo-yo effect). To avoid the negative effects of these pathological behaviors, we rely on the PDCGM (Gondzio and Munari, 2015; Gondzio et al., 2013, 2016). This method is a variant of the standard CG which uses an interior-point algorithm to obtain stable and well-centered dual solutions in the

feasible region of the RMP.

Given a primal-dual feasible, possibly not optimal, solution $(\bar{\lambda}, \bar{u}, \bar{w})$ of the RMP, both a lower and upper bound to the optimal solution of the RMP can be obtained by using the primal and dual values of the objective function as follows:

$$\begin{aligned} Z_{LB}(\bar{\lambda}) &= \sum_{v=1}^{V} \left(\sum_{q \in Q_v} c_{vq} \bar{\lambda}_{vq} \right) \\ Z_{UB}(\bar{u}, \bar{w}) &= b^T \bar{u} + \sum_{v=1}^{V} \bar{w}_v \end{aligned}$$

By assuming the solution is not optimal, we have that $Z_{UB}(\bar{u}, \bar{w}) > Z_{LB}(\bar{\lambda})$. This solution is called ϵ-optimal if it satisfies:

$$Z_{UB}(\bar{u}, \bar{w}) - Z_{LB}(\bar{\lambda}) \leq \epsilon(|Z_{LB}(\bar{\lambda})|)$$

for some $\epsilon > 0$. The primal-dual interior point method provides well centralized dual solutions, in the sense that the complementary products are kept within a vicinity of the central path from the centroid to the optimal solution. More explicitly, a point is well centralized if it satisfies:

$$\gamma\mu \leq (c_{vq} - \bar{u}^T a_{vq} - \bar{w}_v)\lambda_{vq} \leq \frac{1}{\gamma}\mu, \forall v \in V, \forall q \in \bar{Q}_v$$

where $\gamma \in (0, 1)$, a_{vq} is the column of coefficients of the extreme point q, and μ is the barrier parameter that defines the central path for the interior point method. The PDCGM dynamically adjusts the tolerance used to solve the RMP by initially setting a loose value and tightening as it approaches optimality. Algorithm 2 describes the PDCGM.

Algorithm 2 Primal-Dual Column Generation Method

```
procedure PDCGM(ε_max > 0, D > 1, δ > 0)
    LB = −∞, UB = ∞, gap= ∞, ε = 0,5
    while gap ≥ δ do
        Find a centralized primal-dual ε-optimal solution (λ̄, ū, w̄) of the RMP
        LB = max{LB, Z_LB(λ̄)}
        Solve subproblems with values (ū, w̄)
        UB = min{UB, Z_UB(ū, w̄) + Z_SP(ū, w̄)}
        gap = (UB − LB)/(10^−10 + |LB|)
        ε = min{ε_max, gap/D}
        if Z_SP(ū, w̄) > 0 then
            add a column to the RMP
        end if
    end while
end procedure
```

Tolerance for optimality (ϵ) is updated at each iteration using the relative gap (*gap*) between the upper and lower bound. The smaller the relative gap the smaller the intended distance to optimality is as stated in $\epsilon = \min\{\epsilon_{max}, \text{gap}/D\}$. ϵ_{max} is an upper bound that keeps the optimal solution not far away from the optimum and D is the degree of optimality which relates the tolerance ϵ to the relative gap at each iteration. Finally, δ establishes a termination condition based on the relative gap.

5.3 Longest Path algorithm for solving the pricing problem

The subproblem (5.5)–(5.8) is a maximum flow cost problem, and can be solved by the network simplex method. However, by appropriately modifying the network, it can be solved more efficiently with longest path algorithms.

The space-time network defined by sets N and T gives rise to the nodes $(i,t) \in N \times T$ (see Fig. 5.1). The arcs (i,j,t) have two incident nodes, the tail or departing node (i,t) and the head or destination node $(j, t+\tau_{ij})$. Some destination nodes are not defined explicitly by the set $N \times T$ as they go beyond the end of the planning horizon, i.e., $t + \tau_{ij} > T$ (see Fig. 5.2). In order to use a longest path algorithm there needs to be a graph with all its arcs incident to explicit nodes. Next, an extension of the network is described:

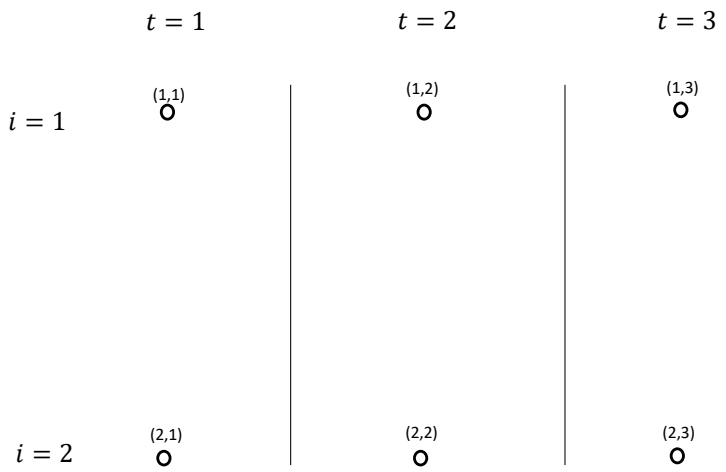


Figure 5.1: Nodes of the network defined by sets N and T.

1. Define the new set $T' = T \cup \{|T|+1\}$. This last period $|T|+1$ contains all nodes beyond the planning horizon (i.e. $t > T$).

2. For each (i,j,t) such that $t + \tau_{ij} > T$, add an arc $((i,t),(j,|T|+1))$.

3. Add a node n_F and an arc $((i,|T|+1),n_F), \forall (i,|T|+1)$. These arcs have a cost 0.

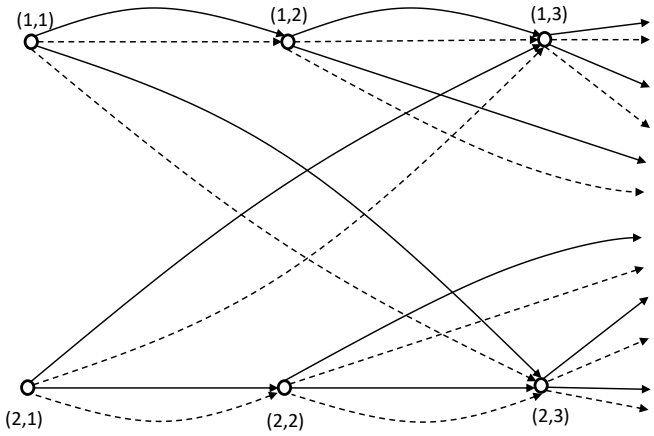


Figure 5.2: Network representation.

Figures 5.3 and 5.4 illustrate an example of the extension in a graph with $N = 2, T = 3$ and travel times: $\tau_{11} = \tau_{22} = 0, \tau_{12} = \tau_{21} = 2$.

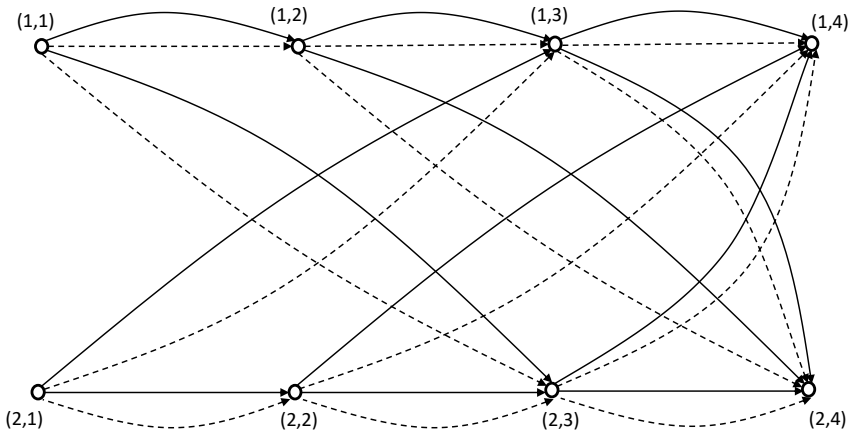


Figure 5.3: Graphic representation of Steps 1 and 2.

This extension of the network enables us to make an equivalence between the solution of the maximum cost flow problem and the aggregated solution of multiple longest path problems. By solving the maximum flow cost problem, where the supply of node n_F equals the sum of all vehicles of type v entering the system within the planning horizon (i.e. $-\sum_{v\in V, i\in N, t\in T}(m_{itv}|m_{itv} > 0)$), the resulting solution equals that of subproblem (5.5)–(5.8). For instance, suppose two vehicles of type 1 enter the system at the pair terminal-period $(1, 1)$ and $(2, 2)$, i.e., $m_{111} = 1$ and $m_{221} = 1$, then the divergence (supply) of node n_F is equal to 2. This feasible solution has the structure of an in-rooted tree (see Fig. 5.5). The equivalence relation between an in-rooted tree and the aggregation of several paths with a common destination is established in Rockafellar (1998), declared as follows:

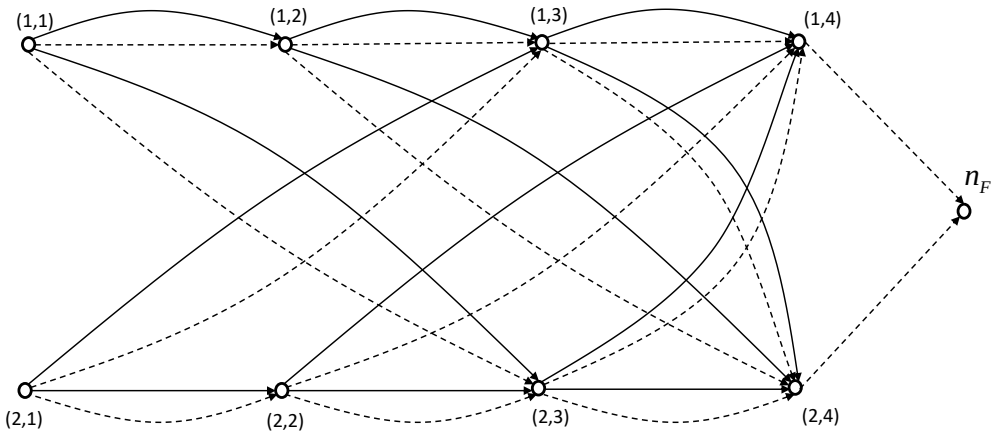


Figure 5.4: Graphic representation of Step 3.

Lemma 5.3.1. *Let V be a set of vehicle types, S_v a set of supply nodes for vehicle type $v \in V$ (i.e. all nodes (i,t) such that $m_{itv} > 0$ for some v), n_F a destination node, Y_{v,n_F} a tree-flow vector with destination n_F and X_{v,s,n_F} a path-flow vector going from $s \in S_v$ to n_F. If Y^{v,n_F} is feasible for the maximum cost flow problem with several origins and one destination, then there exist values for $X_{v,s,n_F}, \forall s \in S_v$ e $v \in V$, such that*

$$Y_{v,n_F} = \sum_{s \in S_v} X_{v,s,n_F}$$

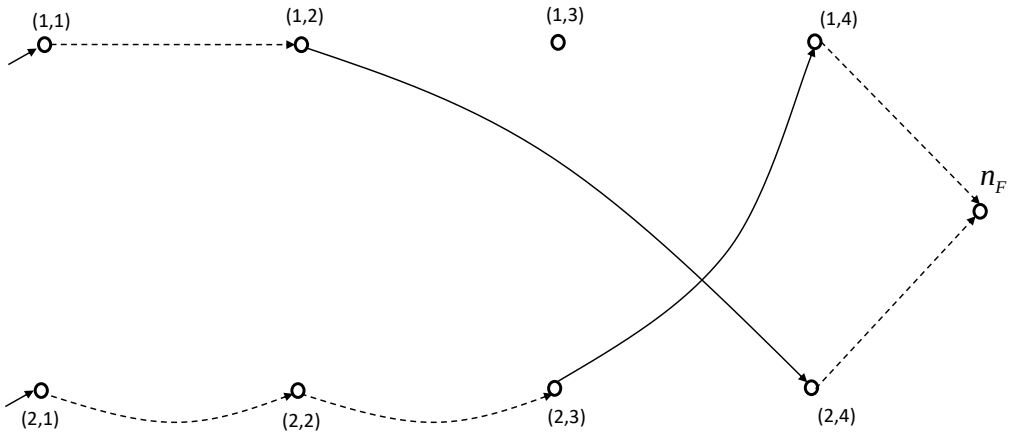


Figure 5.5: Representation of the tree.

5.3.1 Longest Path Algorithm for Directed Acyclic Graphs (DAG)

As the arcs of the graph can have non-negative costs (when $p_{ijv} > u_{ijt}$), it is not possible to use Dijkstra's algorithm for maximization problems (equivalent to not being able to use Dijkstra's algorithm when graphs have negative cost arcs in minimization/shortest-path problems). Nevertheless, given that

the duration of the arcs is strictly positive ($t + \tau_{ijv} > 0, \forall i, j \in N$), the graph is acyclic in nature (Rockafellar, 1998) and we can use a label-setting algorithm tailored for acyclic graphs.

This algorithm's preprocessing procedure requires finding a topological order. For simplicity's sake, let $N^T = (N \times T^{'}) \cup n_F$ and $A^T = (k, k^{'})$, such that, $k \leftrightarrow (i, t)$, $k^{'} \leftrightarrow (j, t + \tau_{ijv})$, and $k, k^{'} \in N^T$. A topological order is a partial order of the nodes with the following property: $f(k), k \in N^T$ such that for each arc $(k, k^{'}) \in A^T$, it holds that $f(k) < f(k^{'})$. Figures 5.6 and 5.7 show the topological order from the above example. From Figure 5.6, it can be observed that each node has a label between 1 and 13, and the label of the tail is less than the label of the head for each arc. If the nodes are ordered according to these new labels, it can be seen from Figure 5.7 that the direction of all arcs is unique (from left to right). Once there is a topological order on the nodes, the longest path algorithm consists in going through all nodes in this order and update the node's distances of the adjacent's incumbent node. The graph for each subproblem $v \in V$ can be different on account of constraints (5.7), however, they are all topologically ordered subgraphs of the graph containing $A_{ijv} = 1, \forall i, j \in N, \forall t \in T, \forall v \in V$. This property can be easily enforced on the optimal solution of the subproblems by penalizing the cost of arcs $A_{ijv} = 0$. In Section 5.3.2, an illustrative example is presented on how to find the longest path of a graph.

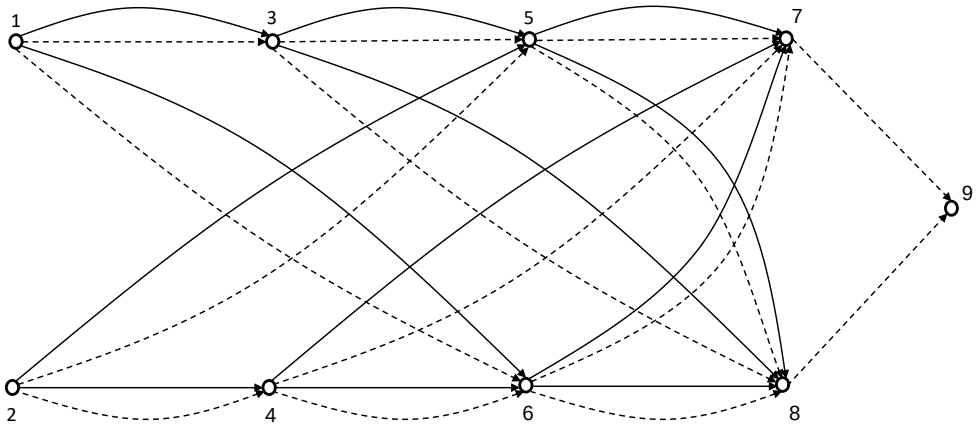


Figure 5.6: Graphic Representation of the topological order for the illustrative example - 1.

Algorithm 3 is the pseudocode of a label-setting algorithm for DAGs (Ahuja et al., 1993). The values $w^x_{k,k'} = p_{k,k'v} - u_{k,k'}$ and $w^y_{k,k'} = c_{k,k'v}$ are the updated costs of moving loaded and moving empty of the pricing subproblem.

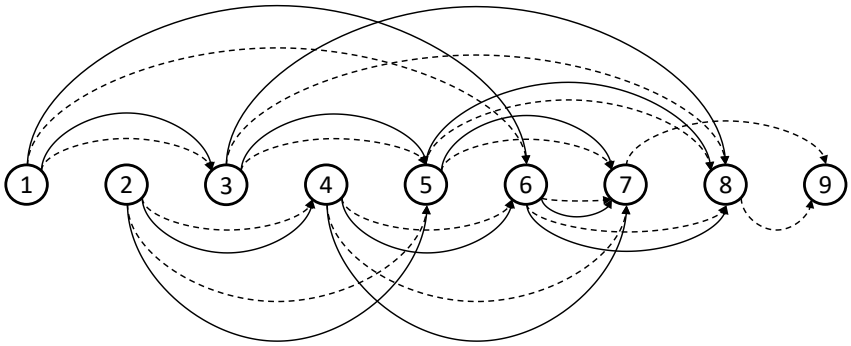


Figure 5.7: Graphic Representation of the topological order for the illustrative example - 2.

Algorithm 3 Longest path algorithm for DAGs

1: $L = TopologicalSort(f(k), k \in N^T)$
2: **procedure** DAG(s, n_F)
3: Initialize $d[k] = -\infty$ and $p[k] = 0, \forall k \in N^T \backslash \{s\}$; $d[s] = 0$
4: **for** $k \leftarrow 1, |N^T|$ **do**
5: **for** $k^{'} \in N^T$ such that$(k, k^{'}) \in A^T$ **do**
6: let $w^x_{k,k'} = p_{k,k'v} - u_{k,k'}$ and $w^y_{k,k'} = c_{k,k'v}$ be the cost of arc $(k, k^{'})$ for vehicle $v \in V$
7: **if** $d[k^{'}] > \max\{d[k] + w^x_{k,k'}, d[k] + w^y_{k,k'}\}$ **then**
8: $d[k^{'}] \leftarrow \max\{d[k] + w^x_{k,k'}, d[k] + w^y_{k,k'}\}$
9: $p[k^{'}] \leftarrow k$
10: **end if**
11: **end for**
12: **end for**
13: **end procedure**

5.3.2 Illustrative example for the Longest Path Problem

To facilitate the understanding of the longest path algorithm, an illustrative example is presented based on the example of Section 5.3.1. The described maximum path is between vertices $f((1,1)) = 1$ and n_F. Table 5.1 shows the arc's costs of loaded and empty trips. Initially, the precedence of all vertices is empty and the label (distance) of vertex 1 equals 0 and the label of the remaining vertices equals $-\infty$. The bold circles represent the incumbent node at each iteration of the algorithm (Line 4), and the shaded circles connected to the bold circles represent the vertices whose labels are updated according to the optimality condition (Lines 7 and 8). For instance, the labels at iteration 3 (Figure 5.10) corresponding to vertices 5 and 8 are updated according to functions: d[5]=max$\{-\infty, 0+0, 0-\infty\}$ and d[8]=max$\{-\infty, 0-1, 0-2\}$, where the first term corresponds to the previous label and the second and third term correspond to the distance when going through vertex 3 and taking the empty and loaded arcs, respectively, from 3 to the forward adjacent node. The following are the graphic representations of the initialization phase,

iteration 1 and 3, and the longest path between the first and last node of the illustrative example. The whole procedure is shown in Appendix A.

Table 5.1: Arcs' costs for the ilustrative example

Arc	1-3	1-6	2-4	2-5	3-5	3-8	4-6	4-7	5-7	5-8	6-7	6-8	7-9	8-9
Empty cost	0,00	-2,50	0,00	-1,50	0,00	-1,00	0,00	-2,20	-2,00	-1,00	0,00	0,00	0,00	0,00
Loaded cost	-inf	2,00	-inf	-2,50	-inf	-2,00	-inf	1,50	2,00	2,50	-inf	-inf	-inf	-inf

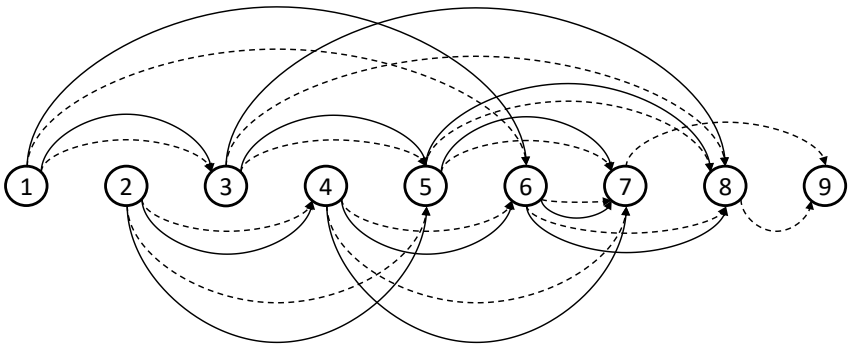


Figure 5.8: Initialization of Maximum Path

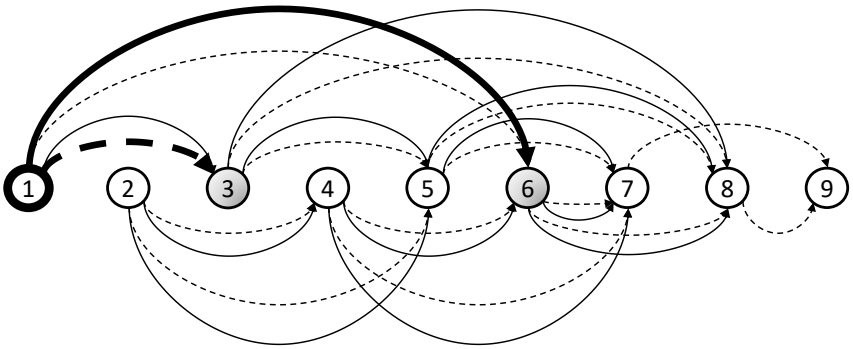


Figure 5.9: Updating labels of vertex 1

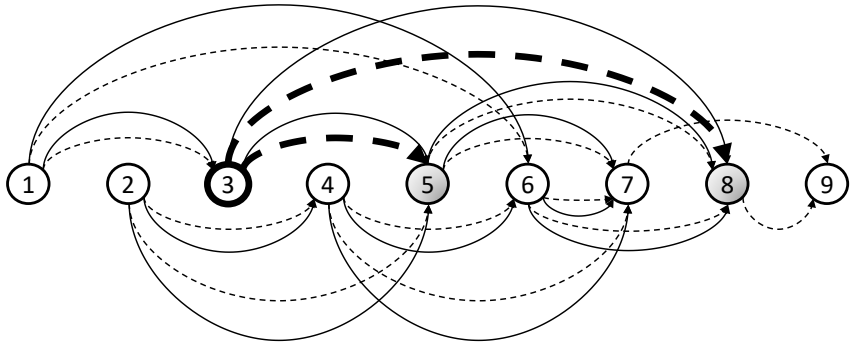


Figure 5.10: Updating labels of vertex 3

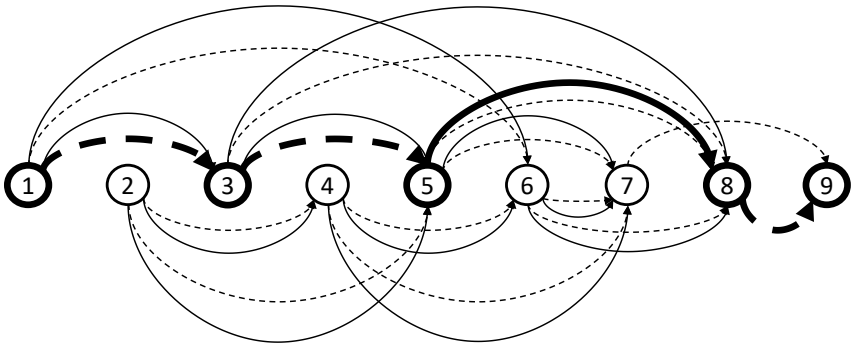


Figure 5.11: Final path between nodes 0 and 9

5.4 Reducing each problem with aggregated fleet to a disaggregated fleet

In this section, we present some results from Cruz (2017) to give evidence of the advantages in using a disaggregated fleet. The experiment uses realistic problem instances with 53 terminals, 36 periods and 130 vehicles, from the case study of Vasco (2012). In this case study, the authors proposed several integer programming models that incorporate real features of a typical Brazilian logistics operator: restriction on vehicle movement, fleet sizing and outsourcing decisions, terminal capacity and backlogged demand. These models were validated through a structured questionnaire made to several professionals in the area of the study's partner-company. The instances were grouped into 18 classes of 30 instances each, with one class for each $|V| = 1..17, 130$, resulting in 18 x 30 = 540 instances. Note that the first class ($|V| = 1$) considers that the 130 vehicles are of the same type, while the last class ($|V| = 130$) considers each vehicle being a unique type of vehicle, that is, a total disaggregated fleet.

Table 5.2 summarizes the results of the following solution approaches applied to each group of 30 instances: (i) solving the LP relaxation of (2.1)–(2.5) using a general-purpose LP solver; (ii) using PDCGM to solve the MP (5.1)–(5.4), together with the longest path algorithm described in Section 5.3.1 for the subproblem; and (iii) applying a simple MIP-heuristic to the last RMP solved by PDCGM – this heuristic consists in imposing integrality to the lambda variables of the RMP (i.e. $\lambda_{vq} \in \mathbb{Z}_+$) and solving the resulting ILP model by a general-purpose ILP solver. The header's columns refer to: the name describing the parameters of the instances in each group (number of terminals - number of periods - number of vehicles - number of types of vehicles - number of requests); the average computational time for solving the LP relaxation of (2.1)–(2.5) using CPLEX (CPU LP); the average computational time for solving the MP (5.1)–(5.4) using PDCGM (CPU PDCGM); and the average relative gap between the feasible solution value obtained by the MIP-heuristic and the optimal value, in percentage (GAP). The relative gap for each instance is calculated as $(f_o - f_h)/f_o$, where f_o is the optimal value of the instance

and f_h is the value of the solution obtained by the MIP-heuristic. A gap of 100% in the fourth column means the best feasible solution found by the MIP-heuristic is the trivial solution of leaving all vehicles idle from when they appear until the end of the planning horizon.

The results in Table 5.2 indicate that the more disaggregated the fleet, the lesser the computational times for the PDCGM and the better the quality of the solutions obtained by the MIP-heuristic. It is worth mentioning that for the instance group 53-36-130-130-300, the LP relaxation of model (2.1)–(2.5) have 26,292,240 variables and 349,164 constraints, which poses excessive computer memory usage and processing difficulties for general-purpose optimization software. As a result, CPLEX could not solve the LP relaxation. On the other hand, the MP model was easily solved by PDCGM and the MIP-heuristic was successful in finding optimal solutions for all instances in this group.

Table 5.2: Computational Times LP - IP - PDCGM for the VAP

Instances	CPU LP (sec)	CPU PDCGM (sec)	GAP
53-36-130-1-300	1,16	2602,20	100.0%
53-36-130-2-300	5,48	1401,50	89.1%
53-36-130-3-300	12,79	784,50	55.6%
53-36-130-4-300	23,32	581,22	33.3%
53-36-130-5-300	39,05	453,50	22.8%
53-36-130-6-300	53,48	348,22	16.0%
53-36-130-7-300	71,19	255,43	11.4%
53-36-130-8-300	85,78	109,96	8.2%
53-36-130-9-300	104,15	174,36	5.6%
53-36-130-10-300	124,30	164,31	4.3%
53-36-130-11-300	265,35	158,80	2.6%
53-36-130-12-300	284,99	130,72	2.5%
53-36-130-13-300	275,99	120,47	1.7%
53-36-130-14-300	291,10	106,63	1.3%
53-36-130-15-300	295,40	102,91	0.7%
53-36-130-16-300	297,79	91,45	0.7%
53-36-130-17-300	270,41	83,61	0.4%
53-36-130-130-300	*	28,91	*

Source: Cruz (2017).

These results are in accordance with Jones et al. (1993), who studied the impact of the formulation on the column generation for a related problem (MCNFP), the disaggregated formulations, that is, the ones where solutions to pricing problems are trees with few leaves like a path, have a better performance in terms of the number of iterations (number of times that the RMP is solved) until reaching optimality. This is due to the fact that the cardinality of extreme points generated in the RMP is smaller in the disaggregated case than in the aggregated one. Furthermore, in the aggregated case it is harder to find vehicles' paths belonging to the optimal solution as they are part of more complex structures, trees, which can make it infeasible to have a given column containing that optimal path. For instance, suppose path 1-4-6 in Figure 5.12 is the optimal path for a given single vehicle (upper-left graph). If

more vehicles are considered within the subproblem as in the other graphs of Figure 5.12, then we can have several optimal solutions containing path 1-4-6 with the inconvenience that other paths can make that solution infeasible.

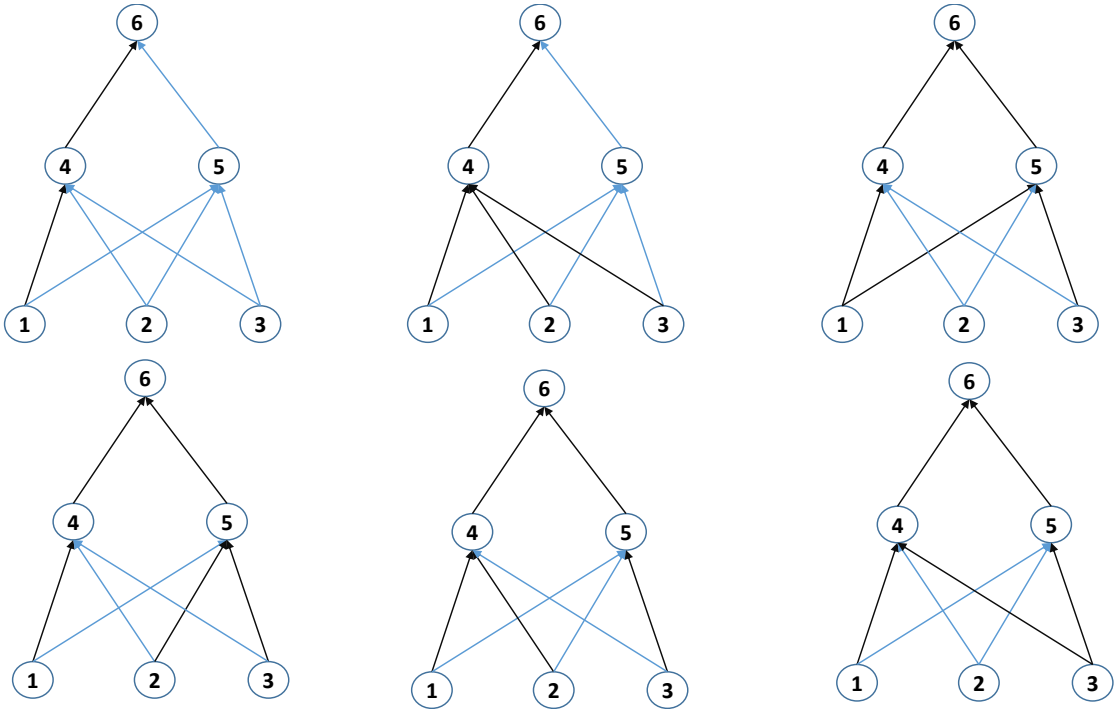


Figure 5.12: Trees containing path 1-4-6.

Another advantage of using the total diaggregated fleet is that is easier to deal with the negative supply m_{itv}. In this case, we do not need to modify the graph as described in Section 5.3; instead, we just order the graph topologically and calculate the longest path between the positive supply node and the negative supply node as shown in Figure 5.13.

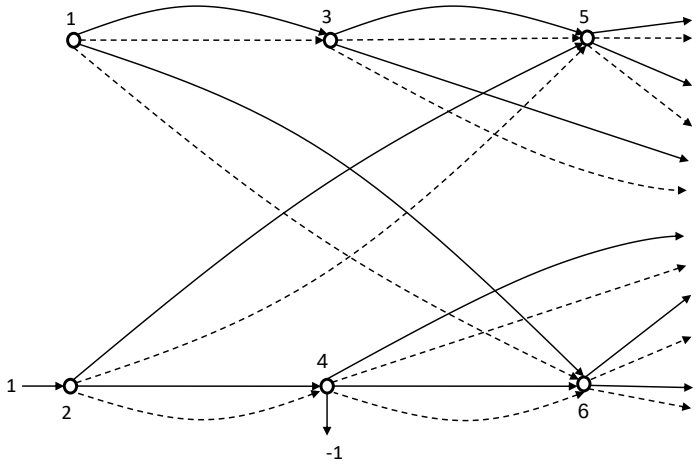


Figure 5.13: Longest path with negative supply.

With this in mind, it can be seen that, without loss of generality, any VAP problem with aggregated types of vehicles can be turned into the total disaggregated VAP case, i.e. $\sum_{i \in N, t \in T, v \in V} m_{itv} = |V|$, with the advantage of having a more efficient algorithm for solving the pricing problem, in addition to columns that hold better information about the optimal solution regarding the MP. For this reason, the next section aims to explore an exact method based on branch-and-bound (BB) for proving optimality in the disaggregated case, that is, each vehicle is its own type of vehicle.

5.5 Branching

Because the VAP was modeled as a binary ILP, in order to obtain an optimal solution, a BB procedure will be needed. However, applying a standard BB procedure to the final RMP of the column generation will not guarantee an optimal (or even feasible) solution, thus columns will need to be generated in each node of the BB (Barnhart et al., 1998). This method is called Branch-and-Price, and we have implemented three procedures for branching in the present study.

When having to decide the next node to be processed along the search tree, we used the best-first search, i.e, processing the node with the best dual bound (maximum dual bound in maximization problems).

5.5.1 Branching on set of arcs

This procedure uses the branching rule proposed by Barnhart et al. (2000), which is based on the arc flow binary variables x_{ijtv} and y_{ijtv} of formulation (2.1)-(2.5) and is compatible in keeping the structure of the pricing problem along the nodes of the branch-and-bound tree. This branching rule consists in finding two fractional paths q_1 and q_2, for a given vehicle $v \in V$. Since the case of a total disaggregated fleet is the one being considered, then these two paths share a node in common from which both of them split, as shown in Figure 5.14.

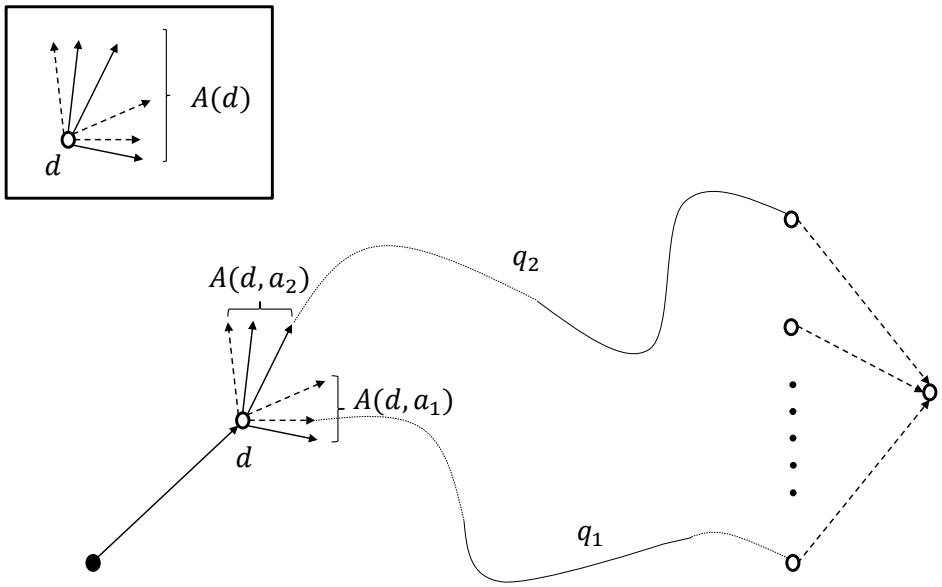


Figure 5.14: Branching rule from Barnhart et al. (2000)

After finding the splitting node for both paths of vehicle v, called the divergence node (d), two sets of arcs are formed out of the outgoing arcs of the divergence node $A(d)$: $A(d, a_1)$ and $A(d, a_2)$. Set $A(d, a_1)$ contains the splitting arc a_1 from path $\lambda_{v_1 q}$ and some other arcs. In the same way, set $A(d, a_2)$ contains the splitting arc a_2 from path $\lambda_{v_2 q}$ and some other arcs. All outgoing arcs different from a_1 and a_2 are evenly divided between both sets with the intuition of keeping a balanced search tree. The two subproblems are created by imposing the following constraints on each of the newly created nodes. For the first:

$$\sum_{q \cap A(d,a_1) \neq \emptyset} \lambda_{vq} = 0$$

and for the second:

$$\sum_{q \cap A(d,a_2) \neq \emptyset} \lambda_{vq} = 0$$

In other words, on the first branch, no path from vehicle v is allowed to use any arc in set $A(d, a_1)$, whereas on the second branch and for the same vehicle, no path is allowed to use any arc in set $A(d, a_2)$. This resulting division is valid as: 1) it prohibits a fractional solution of the LP on each branch; the same divergence node could appear down the tree, however with different fractional paths, and 2) there is a finite number of branches because the number of arcs and vehicles is finite. Each node of the branch-and-bound tree is formed by inheriting only the columns that does not contain any prohibited arc generated in the branching scheme. The subproblem is easily modified by penalizing the costs of

these newly generated prohibited arcs. The branching selection is done, randomly among vehicles with fractional paths, by taking the greatest fractional paths when there are more than two fractional paths for the same vehicle.

5.5.2 Branching on the original variables

Now consider the case where branching is performed on the original variables $x_{ijtv} \in [0,1]$, $y_{ijtv} \in [0,1]$. Suppose branching is performed on some x_{ijtv} and we try to impose $x_{ijtv} = 0$ on one branch and $x_{ijtv} = 1$ on the other branch. The case where $x_{ijtv} = 0$ is easily enforced by penalizing the variable's cost when solving the longest path problems. The case where $x_{ijtv} = 1$ has the drawback of having to deal with multiple, sequential arcs along the planning horizon. For instance, when branching in the root node, enforcing $x_{ijtv} = 1$ entails solving two longest path problems instead of one, as follows: the first going from the origin to (i,t) and the second going from $(j, t+\tau_{ij})$ to n_F. When branching in other nodes rather than the root, to enforce several variables to 1 (i.e., $x_{ijtv} = 1$ or $y_{ijtv} = 1$), we have to deal with the additional problem of sequencing several longest paths among these arcs. This problem of sequencing several longest path is different from the single origin-destination longest path problem.

To avoid this difficulty, we use an adaption of the branching procedure of Vanderbeck and Wolsey (1996). Instead of trying to enforce $x_{ijtv} = 0$ and $x_{ijtv} = 1$ directly within the subproblem, at a given node s of the branch-and-bound tree, we apply branching constraints of the following form to the RMP:

$$x_{ijtv} \leq 0 \leftrightarrow \sum_{q \in Q^v} \bar{x}_{ijtvq} \lambda_{vq} \leq 0 \quad \forall (i,j,t,v) \in UX^s \tag{5.9}$$

$$y_{ijtv} \leq 0 \leftrightarrow \sum_{q \in Q^v} \bar{y}_{ijtvq} \lambda_{vq} \leq 0 \quad \forall (i,j,t,v) \in UY^s \tag{5.10}$$

$$x_{ijtv} \geq 1 \leftrightarrow \sum_{q \in Q^v} \bar{x}_{ijtvq} \lambda_{vq} \geq 1 \quad \forall (i,j,t,v) \in LX^s \tag{5.11}$$

$$y_{ijtv} \geq 1 \leftrightarrow \sum_{q \in Q^v} \bar{y}_{ijtvq} \lambda_{vq} \geq 1 \quad \forall (i,j,t,v) \in LY^s \tag{5.12}$$

where UX^s and UY^s are the set of index tuples related to upper bound branching constraints of the x and y variables, respectively, and LX^s and LY^s are the set of index tuples related to lower bound branching constraints of the x and y variables, respectively. To modify the subproblem and enforce the branching constraints, we use the dual variables θ_{ijtv} of the newly added constraints (5.9)-(5.12) to modify the arc's costs of the subproblem's objective function $Z_{sp(v)}$ as follows:

$$Z_{sp(v)} = \max \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \Big((p_{ijv} - u_{ijt}) x_{ijtv} - c_{ijv} y_{ijtv} \Big) - \sum_{\substack{(i,j,t,v) \in \\ UX^s \cup LX^s}} \theta_{ijtv} x_{ijtv} - \sum_{\substack{(i,j,t,v) \in \\ UY^s \cup LY^s}} \theta_{ijtv} y_{ijtv}.$$

The branching selection is performed by giving priority to the fractional x variables with earlier departing times and choosing randomly among different vehicles to break ties.

5.5.3 Branching on the demand constraints

In relaxed solutions $\bar{\lambda}$, it may be the case that the total demand served by several vehicles at a given arc (i, j, t) is fractional, i.e.,

$$\sum_{v \in V} \sum_{q \in Q_v} \bar{x}_{ijtvq} \bar{\lambda}_{vq} \notin \mathbb{Z}_+. \tag{5.13}$$

Since the partial demand served at a given arc must be integral, we propose the following branching procedure based on adding branching constraints enforcing the integral demand served:

$$\sum_{v \in V} \sum_{q \in Q_v} \bar{x}_{ijtvq} \lambda_{vq} \geq \left\lceil \sum_{v \in V} \sum_{q \in Q_v} \bar{x}_{ijtvq} \bar{\lambda}_{vq} \right\rceil \quad \forall (i, j, t) \in LD^s,$$

$$\sum_{v \in V} \sum_{q \in Q_v} \bar{x}_{ijtvq} \lambda_{vq} \leq \left\lfloor \sum_{v \in V} \sum_{q \in Q_v} \bar{x}_{ijtvq} \bar{\lambda}_{vq} \right\rfloor \quad \forall (i, j, t) \in UD^s,$$

where $\lceil\ \rceil$ and $\lfloor\ \rfloor$ represent the ceiling and floor functions for rounding fractional numbers. UD^s and LD^s are the set of upper bound and lower bound branching constraints. As in the previous branching procedure, the subproblem's objective function $Z_{sp(v)}$ needs to be modified through the dual variables γ_{ijt} of the branching constraints as follows:

$$Z_{sp(v)} = \max \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \Big((p_{ijv} - u_{ijt}) x_{ijtv} - c_{ijv} y_{ijtv} \Big) - \sum_{\substack{(i,j,t) \in \\ UD^s \cup LD^s}} \gamma_{ijt} x_{ijtv}.$$

The branching selection is performed by giving priority to the arcs with earlier departing times. It is worth mentioning that different from the other procedures, this branching scheme alone does not guarantee an integer solution, as the summation in (5.13) may be an integer even for a solution $\bar{\lambda}$ with fractional components. Hence, in our implementation, this branching procedure is used hierarchically, followed by branching on set of arcs (Section 5.5.1) or original variables (Section 5.5.2).

5.6 Computational Experiments

In this section, we present the results of computational experiments with the proposed CG approach and the BP method. We analyse their performance with respect to other exact approaches based on solving the ILP model (2.1)–(2.5) and its LP relaxation by general-purpose solvers. Additionally, we verify the impact of each of the three branching procedures to the performance of the BP method. All methods were implemented in C++ using the PDCGM library and the IBM CPLEX Optimization Studio

12.8.1. All experiments reported in this section were run in a PC with CPU Intel®Core i7-4790S 3.20GHz and 16 GB of RAM.

In the experiments, we solve the 30 large-scale instances from Vasco and Morabito (2016b) already introduced in Section 5.4 (i.e., the instances in class 53-36-130-130-300 at the last line of Table 5.2 with all vehicles of different types). They represent realistic data in terms of network size, planning horizon (24 six-hour periods over 6 days), expected number of loads over the planning horizon and fleet size of a typical Brazilian logistics operator (Vasco, 2012). We also solved small- and large-scale realistic-sized instances randomly generated. The purpose of this instance generation was to test the algorithms with instances having positive integrality gap (i.e. optimal value of the ILP problem strictly smaller than the optimal value of the respective LP relaxation) as there were only six instances with this feature in the group of instances of Vasco and Morabito (2016b). These instance were generated using the procedure described in Section 4.2. After running the instance generator, we used the proposed BP method to test for non-zero optimality gap between the LP and IP solutions. These problem instances and the instance generator can be obtained upon request to the authors. We have created 4 branching schemes by combining the branching procedures of Section 5.5, which results in four BP algorithms for trying to solve optimally the integer VAP. The schemes are

- Scheme A uses only the branching procedure of Section 5.5.1 (branching on set of arcs).
- Scheme B uses only the branching procedure of Section 5.5.2 (branching on original variables).
- Scheme C uses branching procedure of Section 5.5.3 (branching on demand constraints) followed by branching procedure of Section 5.5.1 (branching on set of arcs). At a given node we evaluate if there are fractional solutions on demand constraints and on set of arcs. If both are found, we apply the branching on demand constraints.
- Scheme D works the same as Scheme C, however, branching procedure of Section 5.5.3 is followed by branching procedure of Section 5.5.2 (branching on original variables) instead.

Tables 5.3 to 5.8 show the main results obtained by the proposed CG and BP methods, considering different branching procedures, and the LP and ILP solvers of CPLEX. Since we have four branching schemes with different characteristics that affect the performance of the BP method, we solved the 230 instances using each procedure. A time limit of 3600 seconds was imposed on each run. A sign "*" in the tables indicates that CPLEX could not solve or even mount the model due to lack of computer memory. Columns in these tables refer to:

- Instance is the name of the instance (number of terminals - number of periods - number of vehicles - number of requests.)

- LP is the optimal value of the LP relaxation of model (2.1)–(2.5).
- CPU LP (sec) is the time taken by CPLEX to solve each LP instance to optimality.
- IP is the optimal ILP value of model (2.1)–(2.5).
- CPU IP (sec) is the time taken by CPLEX to solve each ILP instance to optimality.
- CPU PD (sec) is the time taken by the PDCGM to solve the MP (5.1)–(5.4).
- CPU BP-α is the time taken by the BP to solve each ILP instance to optimality with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- N.T-α is total number of nodes created with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- N.E-α is total number of nodes explored with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- UB BP-α is the upperbound reached with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- LB BP-α is the lowerbound reached with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- GAP-α is the relative gap between UB BP-α and LB BP-α when using the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.

Tables 5.3 and 5.4 show the results for the 30 large-scale instances from Vasco and Morabito (2016b). We observe that the proposed CG approach based on the MP (5.1)–(5.4) was significantly more efficient than solving the LP relaxation of model (2.1)–(2.5) by CPLEX. Regarding the IP optimal solution, only 6 instances had positive gap (instances with total number of nodes and number of explored nodes larger than 1 in schemes A, B, C and D). It should be noted that scheme D proved optimality of these 6 instances in less than half the computational time limit.

The results for the randomly generated instances are presented in Tables 5.5 to 5.8. As mentioned before, these instances have a positive integrality gap, as the optimal value of their ILP problems are strictly smaller than the optimal values of the respective LP relaxations. Hence, they are used to better test the effectiveness of the proposed branching procedures. First, we analyse the performance of the proposed CG approach based on the MP (5.1)–(5.4) against solving the LP relaxation of model (2.1)–(2.5) by CPLEX. From Tables 5.5 to 5.8, we observe that the CG approach achieved a superior performance in terms of solving instances to proven optimality. It solved the whole subset of instances, while CPLEX could not mount the models of instances from Tables 5.7 and 5.8.

Regarding the ILP approaches, from instances of Tables 5.5 and 5.7, we observe that the branching based on original variables (scheme B) outperforms the branching based on set of arcs (scheme A). Scheme A solved 8 out of 30 instances, while scheme B solved 17 out of 30 in Table 5.5. All these solved

Table 5.3: Results of the 30 instances of class 53-36-130-130-300 from Vasco and Morabito (2016b) with schemes A and B.

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
53-36-130-300-p1	*	*	*	*	32.98	3600.19	393	256	17437.8	17437.6	0.001%	3601.31	299	277	17437.8	17437.6	0.001%
53-36-130-300-p2	*	*	*	*	28.62	28.85	1	1	19260	19260	0.000%	30.14	1	1	19260	19260	0.000%
53-36-130-300-p3	*	*	*	*	31.03	3600.92	379	229	16634.5	16633.8	0.004%	3600.50	341	227	16634.5	16633.8	0.004%
53-36-130-300-p4	*	*	*	*	35.71	35.94	1	1	19560	19560	0.000%	37.74	1	1	19560	19560	0.000%
53-36-130-300-p5	*	*	*	*	28.78	29.01	1	1	18169.2	18169.2	0.000%	30.90	1	1	18169.2	18169.2	0.000%
53-36-130-300-p6	*	*	*	*	32.67	32.90	1	1	19969.4	19969.4	0.000%	34.02	1	1	19969.4	19969.4	0.000%
53-36-130-300-p7	*	*	*	*	31.33	31.56	1	1	19213.8	19213.8	0.000%	32.97	1	1	19213.8	19213.8	0.000%
53-36-130-300-p8	*	*	*	*	29.23	3600.48	407	217	18475.9	18472.6	0.018%	3608.01	395	205	18475.9	18472.6	0.018%
53-36-130-300-p9	*	*	*	*	36.45	36.68	1	1	15371.4	15371.4	0.000%	38.32	1	1	15371.4	15371.4	0.000%
53-36-130-300-p10	*	*	*	*	24.62	24.85	1	1	18344.8	18344.8	0.000%	26.14	1	1	18344.8	18344.8	0.000%
53-36-130-300-p11	*	*	*	*	36.01	3601.90	375	227	16799.9	16799.6	0.002%	183.89	11	11	16799.6	16799.6	0.000%
53-36-130-300-p12	*	*	*	*	34.37	34.60	1	1	22008.4	22008.4	0.000%	36.03	1	1	22008.4	22008.4	0.000%
53-36-130-300-p13	*	*	*	*	37.51	3607.20	417	244	19628.6	19628.2	0.002%	3601.13	365	231	19628.6	19628.2	0.002%
53-36-130-300-p14	*	*	*	*	31.07	31.30	1	1	19616.6	19616.6	0.000%	32.84	1	1	19616.6	19616.6	0.000%
53-36-130-300-p15	*	*	*	*	38.17	38.40	1	1	20673.2	20673.2	0.000%	40.39	1	1	20673.2	20673.2	0.000%
53-36-130-300-p16	*	*	*	*	35.90	36.13	1	1	17796.2	17796.2	0.000%	37.44	1	1	17796.2	17796.2	0.000%
53-36-130-300-p17	*	*	*	*	32.95	33.18	1	1	17345.2	17345.2	0.000%	35.09	1	1	17345.2	17345.2	0.000%
53-36-130-300-p18	*	*	*	*	36.62	3601.01	405	240	17850.7	17849.2	0.008%	360.25	23	23	17849.2	17849.2	0.000%
53-36-130-300-p19	*	*	*	*	37.09	37.32	1	1	18190.6	18190.6	0.000%	39.14	1	1	18190.6	18190.6	0.000%
53-36-130-300-p20	*	*	*	*	36.21	36.44	1	1	20754.4	20754.4	0.000%	38.20	1	1	20754.4	20754.4	0.000%
53-36-130-300-p21	*	*	*	*	44.80	45.03	1	1	16953.2	16953.2	0.000%	46.76	1	1	16953.2	16953.2	0.000%
53-36-130-300-p22	*	*	*	*	36.82	37.05	1	1	18699.2	18699.2	0.000%	39.24	1	1	18699.2	18699.2	0.000%
53-36-130-300-p23	*	*	*	*	33.96	34.19	1	1	21525.6	21525.6	0.000%	35.31	1	1	21525.6	21525.6	0.000%
53-36-130-300-p24	*	*	*	*	26.51	26.74	1	1	18266.2	18266.2	0.000%	28.20	1	1	18266.2	18266.2	0.000%
53-36-130-300-p25	*	*	*	*	33.49	33.72	1	1	17064.8	17064.8	0.000%	35.61	1	1	17064.8	17064.8	0.000%
53-36-130-300-p26	*	*	*	*	29.71	29.94	1	1	20324.4	20324.4	0.000%	31.04	1	1	20324.4	20324.4	0.000%
53-36-130-300-p27	*	*	*	*	26.61	26.84	1	1	20003	20003	0.000%	28.27	1	1	20003	20003	0.000%
53-36-130-300-p28	*	*	*	*	37.70	37.93	1	1	17956	17956	0.000%	39.57	1	1	17956	17956	0.000%
53-36-130-300-p29	*	*	*	*	30.19	30.42	1	1	19074.6	19074.6	0.000%	32.07	1	1	19074.6	19074.6	0.000%
53-36-130-300-p30	*	*	*	*	32.78	33.01	1	1	16464.6	16464.6	0.000%	35.17	1	1	16464.6	16464.6	0.000%
Mean					33.32	747.12					0.001%	526.52					0.001%

Table 5.4: Results of the 30 instances of class 53-36-130-130-300 from Vasco and Morabito (2016b) with schemes C and D.

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
53-36-130-300-p1	*	*	*	*	32.98	49.65	3	3	17437.6	17437.6	0.000%	46.89	3	3	17437.6	17437.6	0.000%
53-36-130-300-p2	*	*	*	*	28.62	28.75	1	1	19260	19260	0.000%	29.10	1	1	19260	19260	0.000%
53-36-130-300-p3	*	*	*	*	31.03	3600.07	399	235	16634.2	16633.8	0.002%	409.09	33	33	16633.8	16633.8	0.000%
53-36-130-300-p4	*	*	*	*	35.71	36.16	1	1	19560	19560	0.000%	36.83	1	1	19560	19560	0.000%
53-36-130-300-p5	*	*	*	*	28.78	29.30	1	1	18169.2	18169.2	0.000%	29.08	1	1	18169.2	18169.2	0.000%
53-36-130-300-p6	*	*	*	*	32.67	32.98	1	1	19969.4	19969.4	0.000%	33.00	1	1	19969.4	19969.4	0.000%
53-36-130-300-p7	*	*	*	*	31.33	31.49	1	1	19213.8	19213.8	0.000%	31.51	1	1	19213.8	19213.8	0.000%
53-36-130-300-p8	*	*	*	*	29.23	126.07	7	7	18472.6	18472.6	0.000%	127.94	7	7	18472.6	18472.6	0.000%
53-36-130-300-p9	*	*	*	*	36.45	37.06	1	1	15371.4	15371.4	0.000%	36.74	1	1	15371.4	15371.4	0.000%
53-36-130-300-p10	*	*	*	*	24.62	24.90	1	1	18344.8	18344.8	0.000%	25.20	1	1	18344.8	18344.8	0.000%
53-36-130-300-p11	*	*	*	*	36.01	46.14	3	3	16799.6	16799.6	0.000%	46.50	3	3	16799.6	16799.6	0.000%
53-36-130-300-p12	*	*	*	*	34.37	34.85	1	1	22008.4	22008.4	0.000%	34.73	1	1	22008.4	22008.4	0.000%
53-36-130-300-p13	*	*	*	*	37.51	80.31	5	5	19628.2	19628.2	0.000%	80.58	5	5	19628.2	19628.2	0.000%
53-36-130-300-p14	*	*	*	*	31.07	31.63	1	1	19616.6	19616.6	0.000%	31.77	1	1	19616.6	19616.6	0.000%
53-36-130-300-p15	*	*	*	*	38.17	38.52	1	1	20673.2	20673.2	0.000%	38.57	1	1	20673.2	20673.2	0.000%
53-36-130-300-p16	*	*	*	*	35.90	36.08	1	1	17796.2	17796.2	0.000%	36.01	1	1	17796.2	17796.2	0.000%
53-36-130-300-p17	*	*	*	*	32.95	33.11	1	1	17345.2	17345.2	0.000%	33.52	1	1	17345.2	17345.2	0.000%
53-36-130-300-p18	*	*	*	*	36.62	82.89	5	5	17849.2	17849.2	0.000%	82.99	5	5	17849.2	17849.2	0.000%
53-36-130-300-p19	*	*	*	*	37.09	37.03	1	1	18190.6	18190.6	0.000%	37.23	1	1	18190.6	18190.6	0.000%
53-36-130-300-p20	*	*	*	*	36.21	35.77	1	1	20754.4	20754.4	0.000%	36.26	1	1	20754.4	20754.4	0.000%
53-36-130-300-p21	*	*	*	*	44.80	44.67	1	1	16953.2	16953.2	0.000%	44.80	1	1	16953.2	16953.2	0.000%
53-36-130-300-p22	*	*	*	*	36.82	36.97	1	1	18699.2	18699.2	0.000%	37.64	1	1	18699.2	18699.2	0.000%
53-36-130-300-p23	*	*	*	*	33.96	33.92	1	1	21525.6	21525.6	0.000%	34.35	1	1	21525.6	21525.6	0.000%
53-36-130-300-p24	*	*	*	*	26.51	27.42	1	1	18266.2	18266.2	0.000%	27.34	1	1	18266.2	18266.2	0.000%
53-36-130-300-p25	*	*	*	*	33.49	33.59	1	1	17064.8	17064.8	0.000%	33.85	1	1	17064.8	17064.8	0.000%
53-36-130-300-p26	*	*	*	*	29.71	30.02	1	1	20324.4	20324.4	0.000%	30.16	1	1	20324.4	20324.4	0.000%
53-36-130-300-p27	*	*	*	*	26.61	27.28	1	1	20003	20003	0.000%	27.10	1	1	20003	20003	0.000%
53-36-130-300-p28	*	*	*	*	37.70	37.69	1	1	17956	17956	0.000%	37.92	1	1	17956	17956	0.000%
53-36-130-300-p29	*	*	*	*	30.19	30.04	1	1	19074.6	19074.6	0.000%	30.23	1	1	19074.6	19074.6	0.000%
53-36-130-300-p30	*	*	*	*	32.78	33.41	1	1	16464.6	16464.6	0.000%	33.68	1	1	16464.6	16464.6	0.000%
Mean					33.33	159.59					0.000%	53.35					0.000%

small instances took less than half the computational time limit. In addition, scheme A and B solved 4 and 10 out of 30 instances, respectively, in Table 5.7. Note that schemes A and B solved less than half the instances presented in the mentioned tables, and even though scheme B performed better than A, some of those solved instances took more than half the time limit to reach optimality. These results are improved when we combine branching procedures as in schemes C and D. From Tables 5.6 and 5.8, we observe that schemes C and D solved 58 out of 60 instances to optimality (except 2: 50-36-130-700 and 50-36-250-700. Note that scheme A was able to solve 50-36-250-700 in only 56.93 seconds). When comparing schemes C and D, we observe that the difference in computational times is relatively small. We consider these differences are due to implementation details between the branching procedures of Section 5.5.1 and Section 5.5.2, given that the number of total nodes and of nodes explored in both schemes are equal (columns 8,9,16 and 17 of Tables 5.6 and 5.8). This assertion holds for the whole set of instances, which suggests the branching procedure of Section 5.5.3 is very effective in splitting polyhedra with fractional values in order to lower the dual bound obtained from the linear relaxation.

Finally, to further verify the performance of the proposed approaches according to different instance characteristics, we randomly generated additional instances having the number of terminals in the ranges [10,19], [30,39], [40,49] and [56,59]. For each of these ranges, the number of vehicles varies in the following way: for [10,19] it increases from 20 to 50 vehicles; for [30,39], from 200 to 250; for [40,49], from 130 to 170; and for [56,59], from 100 to 25. The numbers of periods are 10, 30, 36 and 36, while the numbers of loads are 20, 300, 500 and 700 for each terminal range, respectively. The detailed results of solving these 200 instances with the same approaches used in the previous experiments are presented in Appendix B. To summarize these results, we resort to the performance profiles proposed by Dolan and Moré (2002), briefly described as follows. The performance profile of a method can be defined as the cumulative distribution function for a given performance metric. In our case, we used the computational time to reach optimality as the performance metric. Since we are interested in obtaining an optimal integer solution, the instances whose stopping criterion was due to exceeding the time limit were considered not solved. More specifically, the value $P(\tau)$ for a given method corresponds to the fraction of instances for which that method provides solutions with a computational time within a factor of 2^{τ} of the best computational time. When $\tau = 0$, the value $P(\tau)$ indicates the proportion of instances for which a given method performed the best, i.e., was the fastest; when $\tau \to \infty$, the $P(\tau)$ indicates the proportion of instances that were solved by a given method. We first show the performance profiles regarding our CG approach (PDCGM) and the LP relaxation of model (2.1)–(2.5) (CPLEX). Figure 5.15 shows the effectiveness of PDCGM, as it was the fastest in almost 80% of the instances ($\tau = 0$). In addition, it solved all instances to optimality, while CPLEX solved only 50% of them within the time limit. It is worth noticing from the tables presented in Appendix A that instances

Table 5.5: Results of the small-scale instances for testing the BP with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
20-20-100-200	14473	3.49	14471	16.23	1.48	3599.07	4815	4783	14471.5	14471	0.00%	671.16	989	989	14471	14471	0.00%
20-20-130-200	31331	4.33	31327	19.13	1.51	3599.58	4361	4091	31327.5	31327	0.00%	3599.83	5517	3905	31329.5	31327	0.01%
20-20-150-200	34622.5	5.45	34618	22.10	1.70	3599.37	4623	3406	34622.5	34618	0.01%	56.30	63	63	34618	34618	0.00%
21-20-100-200	2585.5	5.36	2585	19.09	1.75	3599.85	4877	3738	2585.5	2585	0.02%	14.56	15	15	2585	2585	0.00%
21-20-130-200	4056.5	5.01	4056	21.65	2.05	3599.09	4567	3661	4056.5	4056	0.01%	3599.17	4695	3885	4056.5	4056	0.01%
21-20-150-200	27333.5	5.51	27322	23.34	1.88	7.38	7	7	27322	27322	0.00%	175.71	185	185	27322	27322	0.00%
22-20-100-200	4902.5	5.82	4902	23.76	1.96	86.48	95	95	4902	4902	0.00%	13.49	13	13	4902	4902	0.00%
22-20-130-200	12304	5.71	12301	25.56	1.99	3599.01	5275	3282	12304	12301	0.02%	14.68	15	15	12301	12301	0.00%
22-20-150-200	10972.5	7.14	10971	34.33	2.12	3599.65	4279	2882	10972.5	10971	0.01%	3599.46	3841	2530	10972.5	10971	0.01%
23-20-100-200	14133.5	4.65	14132	19.48	1.90	3599.61	4299	3039	14133.5	14132	0.01%	1126.82	1067	1067	14132	14132	0.00%
23-20-130-200	21749.5	6.44	21745	27.30	1.94	387.39	389	389	21745	21745	0.00%	171.37	159	159	21745	21745	0.00%
23-20-150-200	23366	8.00	23358	31.67	2.20	3599.11	4083	2499	23366	23358	0.03%	3599.11	4465	2605	23366	23358	0.03%
24-20-100-200	10157	5.30	10155	24.04	2.37	4.98	3	3	10155	10155	0.00%	22.78	21	21	10155	10155	0.00%
24-20-130-200	3486.5	11.08	3486	42.33	3.51	3599.57	3193	2020	3486.5	3486	0.01%	3600.44	2503	2375	3486.5	3486	0.01%
24-20-150-200	18158	12.68	18156	52.91	3.26	3600.64	4105	2259	18158	18156	0.01%	3600.07	3541	2631	18158	18156	0.01%
25-20-100-200	9336.5	5.20	9335	23.05	2.33	133.04	123	123	9335	9335	0.00%	9.31	7	7	9335	9335	0.00%
25-20-130-200	33906	8.86	33905	38.92	2.64	3599.42	4343	2783	33906	33905	0.00%	13.09	11	11	33905	33905	0.00%
25-20-150-200	32517	9.41	32508	37.72	2.91	3599.39	3765	2074	32517	32508	0.03%	3599.89	3603	2036	32517	32508	0.03%
26-20-100-200	9816	5.49	9815	22.13	2.64	163.62	145	145	9815	9815	0.00%	17.65	15	15	9815	9815	0.00%
26-20-130-200	16895	7.66	16892	33.17	2.76	3599.49	3439	2492	16895	16892	0.02%	9.97	7	7	16892	16892	0.00%
26-20-150-200	3731.5	19.33	3731	67.78	3.94	3600.89	2679	1894	3731.5	3731	0.01%	3600.75	2927	1825	3731.5	3731	0.01%
27-20-100-200	21095.5	7.05	21090	32.24	3.31	3599.80	3471	2688	21094.5	21090	0.02%	5.24	3	3	21090	21090	0.00%
27-20-130-200	8457.5	11.71	8456	55.14	3.95	3600.25	2945	1670	8457.5	8456	0.02%	3599.23	2367	1973	8457	8456	0.01%
27-20-150-200	26709	9.95	26704	44.40	3.29	3599.21	3155	2244	26709	26704	0.02%	536.04	389	389	26704	26704	0.00%
28-20-100-200	3641.5	10.32	3641	42.01	3.23	6.87	3	3	3641	3641	0.00%	6.86	3	3	3641	3641	0.00%
28-20-130-200	25359	8.64	25355	36.32	3.32	3600.63	3067	2188	25359	25355	0.02%	6.23	3	3	25355	25355	0.00%
28-20-150-200	11225.5	11.36	11222	57.47	3.73	3599.85	2835	1717	11225.5	11222	0.03%	3599.45	2349	2172	11225.5	11222	0.03%
29-20-100-200	5995	7.09	5994	31.58	3.69	3599.60	2915	2215	5995	5994	0.02%	3599.60	2641	2462	5994.5	5994	0.01%
29-20-130-200	8979.5	11.48	8978	49.86	3.60	15.50	9	9	8978	8978	0.00%	3599.29	2605	1994	8979.5	8978	0.02%
29-20-150-200	7978.5	10.10	7977	44.10	3.91	3600.50	2779	1796	7978.5	7977	0.02%	3599.24	2181	2025	7978.5	7977	0.02%

Table 5.6: Results of the small-scale instances for testing the BP with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
20-20-100-200	14473	3.49	14471	16.23	1.48	6.63	7	7	14471	14471	0.00%	5.36	7	7	14471	14471	0.00%
20-20-130-200	31331	4.33	31327	19.13	1.51	9.42	9	9	31327	31327	0.00%	7.51	9	9	31327	31327	0.00%
20-20-150-200	34622.5	5.45	34618	22.10	1.70	3.01	3	3	34618	34618	0.00%	2.63	3	3	34618	34618	0.00%
21-20-100-200	2585.5	5.36	2585	19.09	1.75	3.76	3	3	2585	2585	0.00%	3.09	3	3	2585	2585	0.00%
21-20-130-200	4056.5	5.01	4056	21.65	2.05	4.02	3	3	4056	4056	0.00%	3.51	3	3	4056	4056	0.00%
21-20-150-200	27333.5	5.51	27322	23.34	1.88	4.18	3	3	27322	27322	0.00%	3.56	3	3	27322	27322	0.00%
22-20-100-200	4902.5	5.82	4902	23.76	1.96	6.90	5	5	4902	4902	0.00%	5.71	5	5	4902	4902	0.00%
22-20-130-200	12304	5.71	12301	25.56	1.99	4.72	3	3	12301	12301	0.00%	4.04	3	3	12301	12301	0.00%
22-20-150-200	10972.5	7.14	10971	34.33	2.12	7.16	5	5	10971	10971	0.00%	5.47	5	5	10971	10971	0.00%
23-20-100-200	14133.5	4.65	14132	19.48	1.90	3.75	3	3	14132	14132	0.00%	2.99	3	3	14132	14132	0.00%
23-20-130-200	21749.5	6.44	21745	27.30	1.94	3.92	3	3	21745	21745	0.00%	3.02	3	3	21745	21745	0.00%
23-20-150-200	23366	8.00	23358	31.67	2.20	5.87	5	5	23358	23358	0.00%	5.03	5	5	23358	23358	0.00%
24-20-100-200	10157	5.30	10155	24.04	2.37	8.00	5	5	10155	10155	0.00%	6.28	5	5	10155	10155	0.00%
24-20-130-200	3486.5	11.08	3486	42.33	3.51	17.85	9	9	3486	3486	0.00%	13.02	9	9	3486	3486	0.00%
24-20-150-200	18158	12.68	18156	52.91	3.26	12.45	7	7	18156	18156	0.00%	9.43	7	7	18156	18156	0.00%
25-20-100-200	9336.5	5.20	9335	23.05	2.33	7.10	5	5	9335	9335	0.00%	5.93	5	5	9335	9335	0.00%
25-20-130-200	33906	8.86	33905	38.92	2.64	4.65	3	3	33905	33905	0.00%	4.07	3	3	33905	33905	0.00%
25-20-150-200	32517	9.41	32508	37.72	2.91	6.87	5	5	32508	32508	0.00%	6.05	5	5	32508	32508	0.00%
26-20-100-200	9816	5.49	9815	22.13	2.64	4.37	3	3	9815	9815	0.00%	4.16	3	3	9815	9815	0.00%
26-20-130-200	16895	7.66	16892	33.17	2.76	4.20	3	3	16892	16892	0.00%	3.93	3	3	16892	16892	0.00%
26-20-150-200	3731.5	19.33	3731	67.78	3.94	24.24	11	11	3731	3731	0.00%	21.92	11	11	3731	3731	0.00%
27-20-100-200	21095.5	7.05	21090	32.24	3.31	4.93	3	3	21090	21090	0.00%	4.54	3	3	21090	21090	0.00%
27-20-130-200	8457.5	11.71	8456	55.14	3.95	33.61	19	19	8456	8456	0.00%	29.05	19	19	8456	8456	0.00%
27-20-150-200	26709	9.95	26704	44.40	3.29	7.41	5	5	26704	26704	0.00%	6.69	5	5	26704	26704	0.00%
28-20-100-200	3641.5	10.32	3641	42.01	3.23	47.20	29	29	3641	3641	0.00%	42.24	29	29	3641	3641	0.00%
28-20-130-200	25359	8.64	25355	36.32	3.32	5.14	3	3	25355	25355	0.00%	4.63	3	3	25355	25355	0.00%
28-20-150-200	11225.5	11.36	11222	57.47	3.73	6.16	3	3	11222	11222	0.00%	5.49	3	3	11222	11222	0.00%
29-20-100-200	5995	7.09	5994	31.58	3.69	8.81	5	5	5994	5994	0.00%	7.48	5	5	5994	5994	0.00%
29-20-130-200	8979.5	11.48	8978	49.86	3.60	80.44	49	49	8978	8978	0.00%	72.53	49	49	8978	8978	0.00%
29-20-150-200	7978.5	10.10	7977	44.10	3.91	7.37	3	3	7977	7977	0.00%	6.73	3	3	7977	7977	0.00%

Table 5.7: Results of the large-scale instances for testing the BP with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
50-36-100-700	*	*	*	*	23.56	3599.42	527	424	50963.5	50961	0.00%	36.77	3	3	50961	50961	0.00%
50-36-130-700	*	*	*	*	29.24	3600.10	563	316	57063.5	57048	0.03%	3600.59	375	333	57063.5	57048	0.03%
50-36-150-700	*	*	*	*	31.62	40.88	3	3	22655	22655	0.00%	3602.52	477	264	22658.5	22656	0.01%
50-36-180-700	*	*	*	*	34.10	3608.23	469	246	62142	62122	0.03%	3600.63	451	264	62141.5	62122	0.03%
50-36-200-700	*	*	*	*	34.85	3609.31	441	229	28848	28843	0.02%	3607.46	421	242	28848	28843	0.02%
50-36-250-700	*	*	*	*	40.64	56.93	3	3	111444	111444	0.00%	3613.44	367	183	111459	111444	0.01%
51-36-100-700	*	*	*	*	24.03	3356.60	399	399	17676	17676	0.00%	62.19	7	7	17676	17676	0.00%
51-36-130-700	*	*	*	*	30.62	3606.02	545	284	62850	62845	0.01%	2502.91	223	223	62845	62845	0.00%
51-36-150-700	*	*	*	*	30.51	3605.89	497	274	61616	61608	0.01%	3599.87	473	286	61616	61608	0.01%
51-36-180-700	*	*	*	*	37.18	3604.87	423	250	45128.5	45116	0.03%	3609.76	429	229	45128.5	45116	0.03%
51-36-200-700	*	*	*	*	38.92	3605.90	407	217	69131.5	69124	0.01%	3609.21	327	255	69127	69124	0.00%
51-36-250-700	*	*	*	*	49.22	3613.18	319	176	29395.5	29392	0.01%	3602.35	327	172	29395.5	29392	0.01%
52-36-100-700	*	*	*	*	25.76	231.81	27	27	47273	47273	0.00%	79.83	9	9	47273	47273	0.00%
52-36-130-700	*	*	*	*	33.24	3600.56	515	287	60092.5	60081	0.02%	3607.32	551	275	60092	60081	0.02%
52-36-150-700	*	*	*	*	28.20	3607.79	447	273	75999	75988	0.01%	3608.18	355	305	75999	75988	0.01%
52-36-180-700	*	*	*	*	37.01	3609.97	415	225	54700.5	54693	0.01%	3606.44	387	249	54700.5	54693	0.01%
52-36-200-700	*	*	*	*	43.32	3610.06	355	190	33918	33908	0.03%	3610.95	363	194	33918	33908	0.03%
52-36-250-700	*	*	*	*	52.17	3614.09	283	153	8242	8241	0.01%	3604.14	291	154	8242	8241	0.01%
53-36-100-700	*	*	*	*	26.39	1775.47	199	199	18727	18727	0.00%	218.01	23	23	18727	18727	0.00%
53-36-130-700	*	*	*	*	31.03	3600.58	469	297	36395.5	36388	0.02%	3604.30	521	286	36393.5	36388	0.02%
53-36-150-700	*	*	*	*	38.67	3606.57	437	251	63064.5	63060	0.01%	454.83	39	39	63060	63060	0.00%
53-36-180-700	*	*	*	*	38.71	3601.22	395	208	28789	28785	0.01%	3602.49	401	213	28787.5	28785	0.01%
53-36-220-700	*	*	*	*	42.52	3609.63	345	201	101741	101729	0.01%	3612.15	365	182	101741	101723	0.02%
53-36-250-700	*	*	*	*	49.47	3601.82	303	179	104928	104922	0.01%	65.66	3	3	104922	104922	0.00%
54-36-100-700	*	*	*	*	29.48	3606.00	573	301	23558	23551	0.03%	999.09	95	95	23551	23551	0.00%
54-36-130-700	*	*	*	*	33.05	3602.89	349	293	39538.5	39535	0.01%	667.68	61	61	39535	39535	0.00%
54-36-150-700	*	*	*	*	38.63	3601.22	349	216	4722.5	4722	0.01%	3605.55	277	213	4722.5	4722	0.01%
54-36-180-700	*	*	*	*	40.40	3601.80	297	208	36089	36084	0.01%	57.98	3	3	36084	36084	0.00%
54-36-200-700	*	*	*	*	41.92	3600.62	355	195	54219	54198	0.04%	3601.31	333	210	54219	54198	0.04%
54-36-250-700	*	*	*	*	57.90	3615.92	273	144	8532	8531	0.01%	3599.89	213	149	8532	8531	0.01%

Table 5.8: Results of the large-scale instances for testing the BP with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
50-36-100-700	*	*	*	*	23.56	36.61	3	3	50961	50961	0.00%	34.71	3	3	50961	50961	0.00%
50-36-130-700	*	*	*	*	29.24	3606.93	495	306	-57052.5	57048	0.01%	3601.84	525	328	57052.5	57048	0.01%
50-36-150-700	*	*	*	*	31.62	68.46	5	5	22656	22656	0.00%	68.35	5	5	22656	22656	0.00%
50-36-180-700	*	*	*	*	34.10	630.27	57	57	62122	62122	0.00%	627.10	57	57	62122	62122	0.00%
50-36-200-700	*	*	*	*	34.85	38.00	3	3	28843	28843	0.00%	37.93	3	3	28843	28843	0.00%
50-36-250-700	*	*	*	*	40.64	3600.66	329	213	111450	111444	0.01%	3611.77	329	216	111450	111444	0.01%
51-36-100-700	*	*	*	*	24.03	99.38	11	11	17676	17676	0.00%	99.31	11	11	17676	17676	0.00%
51-36-130-700	*	*	*	*	30.62	42.16	3	3	62845	62845	0.00%	42.11	3	3	62845	62845	0.00%
51-36-150-700	*	*	*	*	30.51	171.83	17	17	61608	61608	0.00%	171.96	17	17	61608	61608	0.00%
51-36-180-700	*	*	*	*	37.18	1802.81	153	153	45116	45116	0.00%	1798.33	153	153	45116	45116	0.00%
51-36-200-700	*	*	*	*	38.92	54.47	3	3	69124	69124	0.00%	55.16	3	3	69124	69124	0.00%
51-36-250-700	*	*	*	*	49.22	767.03	43	43	29392	29392	0.00%	772.57	43	43	29392	29392	0.00%
52-36-100-700	*	*	*	*	25.76	38.55	3	3	47273	47273	0.00%	38.97	3	3	47273	47273	0.00%
52-36-130-700	*	*	*	*	33.24	62.32	5	5	60081	60081	0.00%	63.88	5	5	60081	60081	0.00%
52-36-150-700	*	*	*	*	28.20	45.83	3	3	75988	75988	0.00%	46.27	3	3	75988	75988	0.00%
52-36-180-700	*	*	*	*	37.01	206.15	17	17	54693	54693	0.00%	208.32	17	17	54693	54693	0.00%
52-36-200-700	*	*	*	*	43.32	89.17	5	5	33908	33908	0.00%	91.11	5	5	33908	33908	0.00%
52-36-250-700	*	*	*	*	52.17	223.27	13	9	8242	8242	0.00%	224.81	13	9	8242	8242	0.00%
53-36-100-700	*	*	*	*	26.39	147.35	15	15	18727	18727	0.00%	150.27	15	15	18727	18727	0.00%
53-36-130-700	*	*	*	*	31.03	51.23	3	3	36388	36388	0.00%	51.06	3	3	36388	36388	0.00%
53-36-150-700	*	*	*	*	38.67	54.46	3	3	63060	63060	0.00%	55.33	3	3	63060	63060	0.00%
53-36-180-700	*	*	*	*	38.71	1020.99	77	77	28785	28785	0.00%	1023.86	77	77	28785	28785	0.00%
53-36-220-700	*	*	*	*	42.52	106.86	7	7	101729	101729	0.00%	105.90	7	7	101729	101729	0.00%
53-36-250-700	*	*	*	*	49.47	65.29	3	3	104922	104922	0.00%	65.19	3	3	104922	104922	0.00%
54-36-100-700	*	*	*	*	29.48	135.83	13	13	23551	23551	0.00%	137.10	13	13	23551	23551	0.00%
54-36-130-700	*	*	*	*	33.05	45.49	3	3	39535	39535	0.00%	45.46	3	3	39535	39535	0.00%
54-36-150-700	*	*	*	*	38.63	79.96	5	5	4722	4722	0.00%	81.05	5	5	4722	4722	0.00%
54-36-180-700	*	*	*	*	40.40	107.57	7	7	36084	36084	0.00%	107.63	7	7	36084	36084	0.00%
54-36-200-700	*	*	*	*	41.92	44.16	3	3	54198	54198	0.00%	45.06	3	3	54198	54198	0.00%
54-36-250-700	*	*	*	*	57.90	1416.50	59	59	8531	8531	0.00%	1418.64	59	59	8531	8531	0.00%

that were solved faster by CPLEX have number of terminals in the range [10,19], although the difference in computational times within this range do not exceed 1 second.

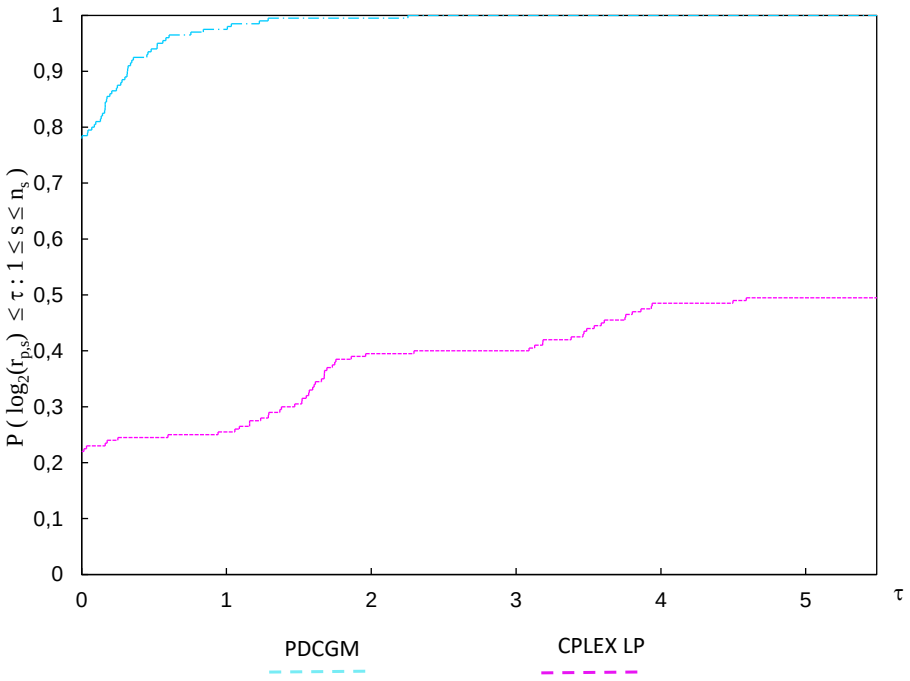


Figure 5.15: Performance profile of PDCGM vs CPLEX.

Regarding the integer optimal solution, Figure 5.16 shows the performance profiles for the branching schemes. We observe that scheme B (blue line) performed better than scheme A (purple line) in terms of solving capabilities, as it solved 50% of the instances compared to 30%; and time efficiency, as it solved 15% of the instances faster compared to 5%. Both schemes A and B are outperformed by C (red line) and D (green line) which give priority to branching on demand constraints. Both schemes, C and D, solved all instances except two (the ones presented in Table 5.8) and they have similar time efficiency performance as evidenced by the closeness of both lines, which was already observed in Tables 5.5 to 5.8. Figure 5.17 compares the performance of CPLEX to the best of the four branching schemes: D. Even though schemes C and D have similar performance (this similarity is mainly due to them sharing the same main branching procedure), we chose scheme D as the best since its subjacent differing branching procedure is more effective (as seen by comparing scheme B to A). We observe that our proposed algorithm outperforms significantly CPLEX in solving the integer VAP. It solved faster almost 80% of the instances, it was able to solve almost 100% of the instances (albeit we still have high quality bounds for the 2 unsolved), while CPLEX could not even mount the model of approximately 50% of the instances.

By and large, from the results shown in this section, it can be seen that the proposed BP is more competitive than the BC from CPLEX when solving large-scale instances. The proposed method with the branching schemes C and D outperformed CPLEX in almost all of the instances with 20 or more

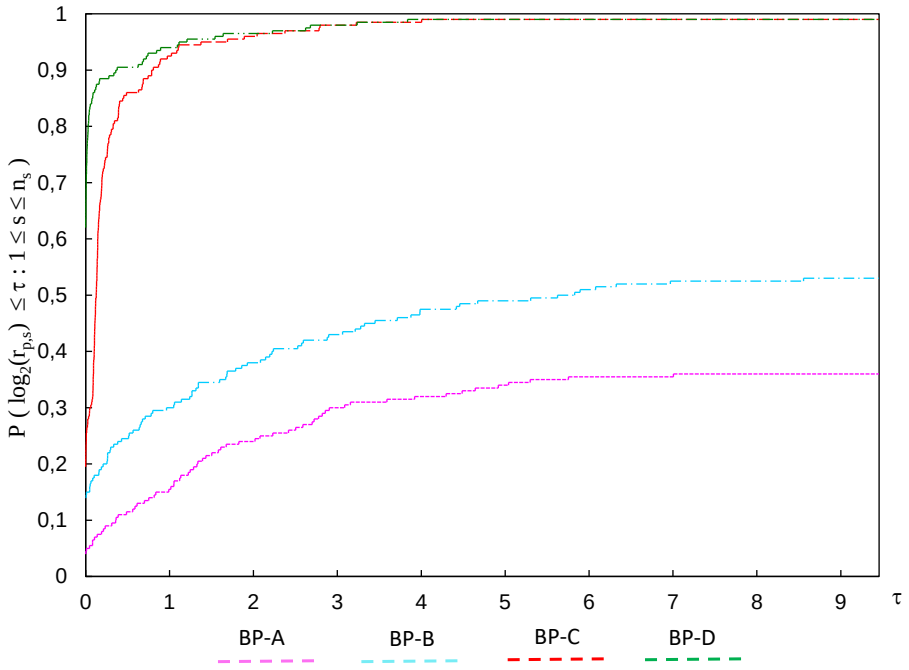


Figure 5.16: Performance profile of the 4 branching schemes.

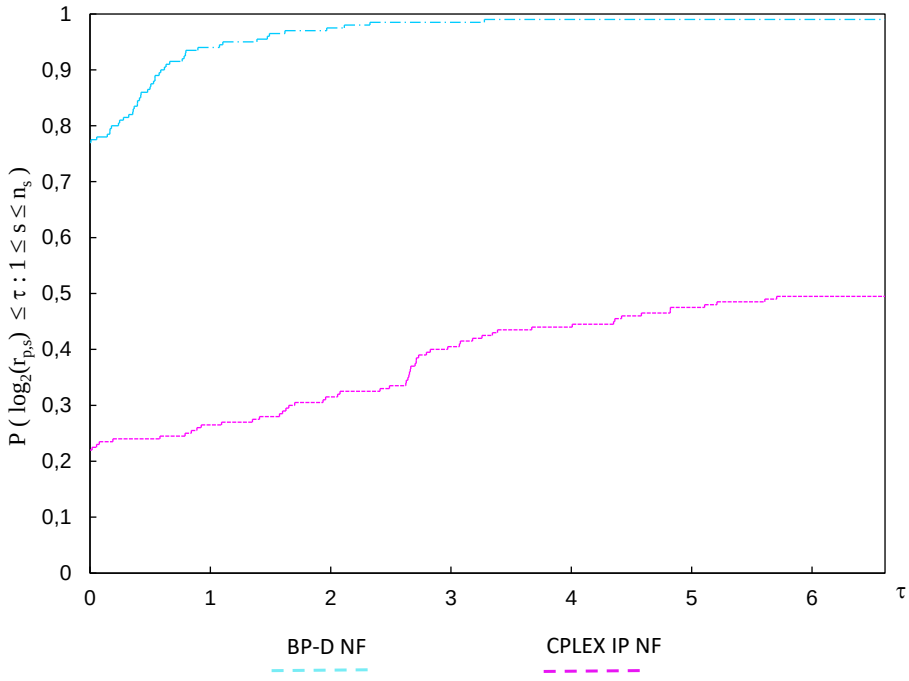


Figure 5.17: Performance profile of CPLEX vs BP-D.

terminals in terms of solving capabilities and time efficiency (except three 28-20-100-200, 29-20-130-200 and 31-30-230-300 which CPLEX solved faster than schemes C and D, see Appendix B). In addition, most instances with 40 or more terminals could not be processed by CPLEX, while they were optimally solved by the proposed method, schemes C and D, within reasonable computational times. These results

clearly show the advantage of using the proposed BP for solving large-scale instances of the VAP.

5.7 Final Considerations

In this chapter, we presented a BP method for the VAP, based on the arc-demand model presented in Section 2.3. The column generation is based on an Interior Point method and the subproblem was found to be reduced to a multiple shortest path problem. We have adapted two branching procedures from the literature and created four branching schemes for solving optimally the integer VAP. Small-scale and large-scale instances were solved with the proposed method and its performance results analyzed and compared to those obtained by the general-purpose solver CPLEX. The BP method with two of the branching schemes turned out to be more effective than CPLEX in solving large-scale instances. The main results of this chapter were published in Cruz et al. (2020).

Chapter 6

A new formulation for the VAP and a Branch-and-Price method

In this chapter, a new formulation for the VAP is proposed where demand for vehicles occur at the nodes of the subjacent graph, hence, we call it the node-demand formulation. The size of this new formulation depends on different parameters than the formulation of Chapter 5. Additionally, we proposed a BP method to optimally solve small and large-scale instances with this new formulation.

6.1 Definition of the node-demand formulation

To introduce the alternative formulation, we consider the notation presented in Section 2.3 and the additional sets and parameters defined as follows. Let R be the set of all requests for freight transportation services along the planning horizon. There is one request $r \in \{1, .., |R|\}$ for each triplet (i, j, t) such that $d_{ijt} > 0, \forall i, j \in N, t \in T$. Each request $r \in R$ is determined by the departing terminal $i_r \in N$, the delivery terminal $j_r \in N$, the starting travel period $t_r \in T$ and the number of vehicles needed to meet the demand $D_r = d_{i_r j_r t_r}$. Based on this, the VAP consists in determining feasible sequences of requests, such that each sequence is executed by a given available vehicle in a way that maximizes the overall profit. In addition, two artificial requests (depots) are added: the 0 depot from which all sequences of requests have to depart and the $|R| + 1$ depot where all sequences of requests arrive at. We use k_v and h_v to denote the location and period, respectively, at which vehicle v appears according to the previous definition m_{itv} used in the arc-demand formulation of Section 2.3. For example, $m_{422} = 1$ is represented as $k_1 = 4$ and $h_1 = 2$.

We then use a *request network* to represent the problem, instead of the space-time network used in the arc-demand formulation. In this request network, there is one node for each request $r \in R$, and one arc (r, s) for each pair of requests $r, s \in R$ such that it is feasible to serve request r and then

s consecutively, using the same vehicle (of at least one type). When the delivery terminal of r is the same as the departing terminal of s (i.e., $j_r = i_s$), then the traversal of arc (r, s) means that the vehicle finished request r and is ready to start request s, without repositioning. Otherwise, the arc traversal corresponds to an empty movement of the vehicle, repositioning from j_r to i_s.

To illustrate the transformation to the new formulation, we introduce a small instance in the time-space network. Figure 6.1 shows an example for $N = \{1, ..., 6\}$ and $T = \{1, ..., 6\}$. In this example, there are four requests for freight transportation services ($d_{422} = 3, d_{562} = 1, d_{233} = 3, d_{515} = 2$) and two vehicles (i.e., $|V| = 2$). Vehicle 1 becomes available at terminal 6 and time period 1 ($m_{611} = 1$) while vehicle 2 becomes available at terminal 4 and time period 2 ($m_{422} = 1$). There is an extra column of terminals ($t > 6$) representing all terminal-period pairs extending beyond the planning horizon. In addition, there is a sink n_f capturing all the flow coming from the extended space-time network, which is helpful is defining the proposed formulation of this section. Figure 6.2 illustrates an optimal solution for this example. In the solution, vehicle 1 follows path $\{(6, 5, 1), (5, 6, 2), (6, 5, 3), (5, 5, 4), (5, 1, 5), (1, 1, 6)\}$ and vehicle 2 follows path $\{(4, 2, 2), (2, 3, 3), (3, 3, 5), (3, 3, 6)\}$.

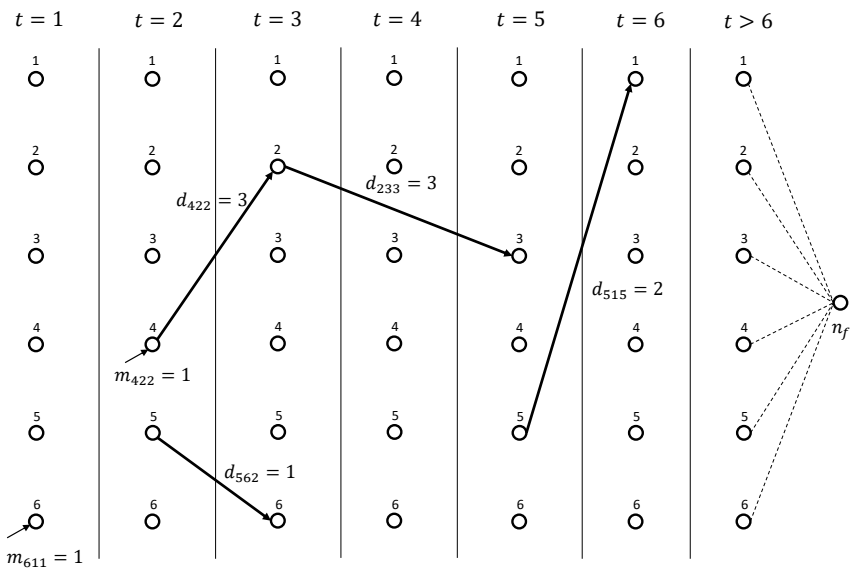


Figure 6.1: The problem's parameters.

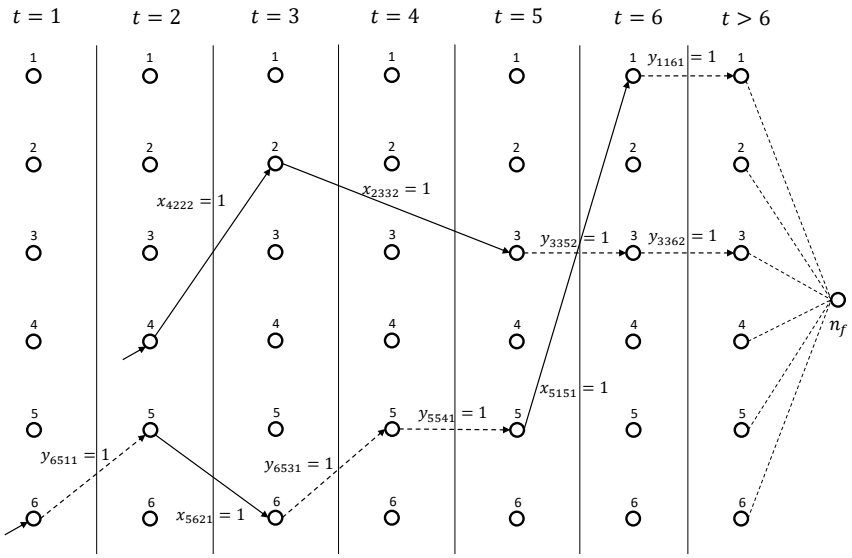


Figure 6.2: An optimal solution for the example.

Figure 6.3 illustrates the request network for the above mentioned example. Circles represent requests whereas triangles represent both artificial depots. The numbers above the circles represent the demand D_r while the triplets within contain the departing terminal $i_r \in N$, the delivery terminal $j_r \in N$ and the starting travel period $t_r \in T$, which will be necessary in the profit/cost definition. Note that by ordering the request nodes according to their starting time period t_r, we inherit the acyclic property from the formulation of Section 2.3. The arcs in this network represent the transition of vehicles either between two requests or between a request and a depot (and viceversa). Note that this transition can represent no event at all when the delivery terminal of r (j_r) is the same as the departing terminal of s (i_s). Finally, in comparison to the space-time network, this alternative network shows a reduced number of nodes and arcs for instances with a moderate number of requests. For example, the 30 node and 216 arc space-time network presented in Figure 6.1 is reduced to a network of only 6 nodes and 15 arcs, totalling 30 decision variables and 18 constraints. Figure 6.4 shows the same optimal solution illustrated in Figure 6.2 but considering the request network.

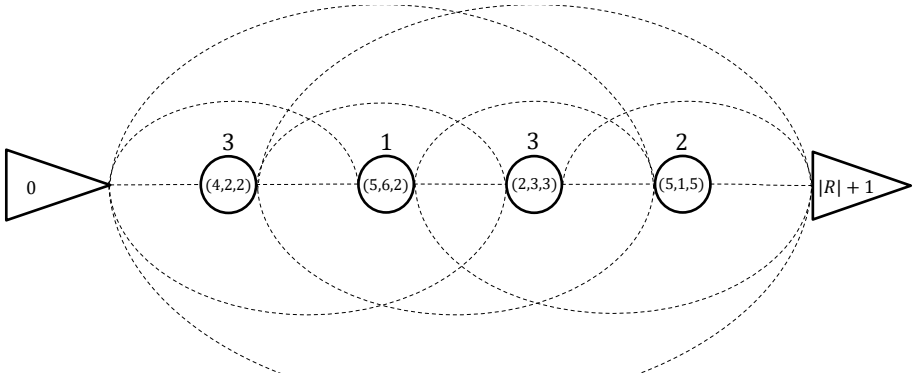


Figure 6.3: Graphic representation of a request network for the same example in Figure6.1.

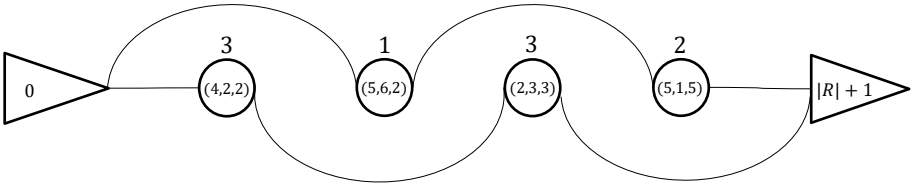


Figure 6.4: An optimal solution in the request network for the same example in Figure 6.2.

In the same way as the demands, profits and costs from vehicles movements have to be redefined in terms of requests. Let P_{rsv} be either the profit or the cost associated to meeting request $s \in R$ consecutively after meeting request $r \in R$ with vehicle $v \in V$. As mentioned before, the traversal of arc (r, s) may require an empty movement of the vehicle, which may shadow the profit related to meet the requests and, hence, $P_{rsv} \geq 0$ corresponds to a profit, or to a cost otherwise. These values are straightforward to compute if the triangle inequality holds for the costs and if we assume that any vehicle can move from a given terminal to any other. However, these are not valid in the VAP because of the restriction of movements. Indeed, if $A_{j_r i_s v} = 0$ for $j_r \neq i_s$, the empty movement may still be possible, but through the shortest path between these terminals, which is a consequence of the fact that optimal solutions for each vehicle $v \in V$ in the time-space network are longest paths since they are maximizing profits (see Lemma 5.3.1 in Section 5.3). Hence, we use the shortest path considering empty costs only of each vehicle type $v \in V$ to calculate the corresponding profits of the following events that can occur between two nodes of the request network: (a) traveling empty between the entering location of the vehicle (k_v) and a request; (b) traveling empty between the delivery terminal of one request and the departing terminal of its consecutive request; (c) traveling empty between the delivery terminal of a request and the end of the planning horizon; and (d) traveling empty between the entering location of the vehicle and the end of the planning horizon. Note that in events (c) and (d) traveling empty is still needed when holding vehicles in inventory is not possible (as is the case in some of the realistic-sized instances considered in this paper).

From the previous definition of possible events between nodes, P_{rsv} is defined in terms of parameters p_{ijv} and c_{ijv} as follows:

$$P_{rsv} = \begin{cases} p_{i_r j_r v} - \text{SP}((j_r, t_r + \tau_{i_r j_r}), (i_s, t_s), v), & \text{if } r, s \in R \\ p_{i_r j_r v} - \text{SP}((j_r, t_r + \tau_{i_r j_r}), n_f, v) & \text{if } r \in R \wedge s = |R| + 1 \\ -\text{SP}((k_v, h_v), (i_s, t_s), v), & \text{if } r = 0 \wedge s \in R \\ -\text{SP}((k_v, h_v), n_f, v), & \text{if } r = 0 \wedge s = |R| + 1 \end{cases}$$

where SP(A, B, v) denotes the shortest path for a vehicle of type v between nodes A and B of the time-space network and arc lengths correspond to empty vehicle costs c_{ijv}. This can be efficiently computed in advance and is used as an input parameter in the model. Whenever a vehicle ends attending a request or enters the system at a given terminal where it is not allowed to be held idle, it is still necessary to compute the shortest path to n_f if there is no request to attend afterwards (Cases 2 and 4 of the profit definition). Figure 6.5 shows the illustrative representation of these events in the request network and their relationship to the definition of P_{rsv}.

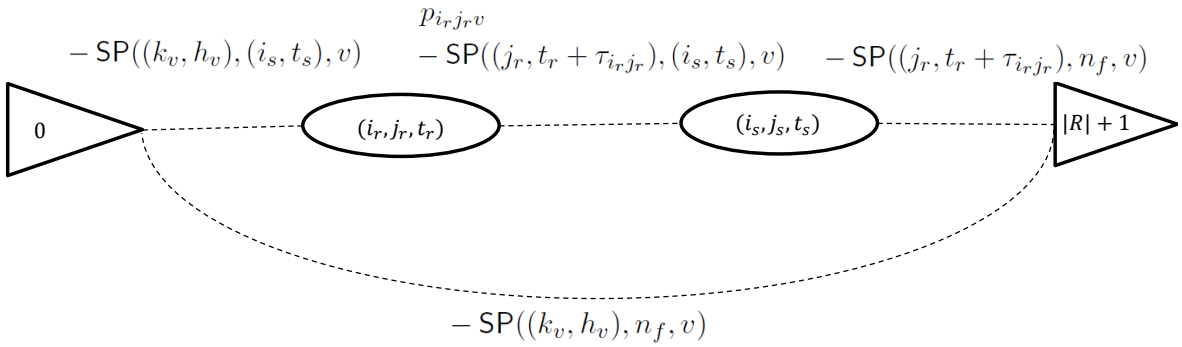


Figure 6.5: Redefinition of profit/cost P_{rsv}

The decision variables in the alternative formulation, defined as x_{rsv}, take the value of 1 if vehicle v serve request s after serving request r, and 0 otherwise. Using the presented definitions, the node-demand formulation of the VAP writes as:

$$\max \quad \sum_{v \in V} \sum_{r \in R \cup \{0\}} \sum_{\substack{s \in S \cup \{|R|+1\} \\ s > r}} P_{rsv} x_{rsv} \tag{6.1}$$

$$\text{s.t.} \quad \sum_{\substack{s \in R \cup \{|R|+1\} \\ s > r}} x_{rsv} - \sum_{\substack{s \in R \cup \{0\} \\ s < r}} x_{srv} = \begin{cases} 1 & \text{if } r = 0 \\ -1 & \text{if } r = |R| + 1 \\ 0 & \text{if } r \neq 0 \wedge r \neq |R| + 1 \end{cases}$$

$$\forall r \in R \cup \{0\} \cup \{|R| + 1\}, \forall v \in V, \tag{6.2}$$

$$\sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} x_{rsv} \leq D_r, \quad \forall r \in R, \tag{6.3}$$

$$x_{srv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R \cup \{|R| + 1\}, \forall v \in V, \tag{6.4}$$

$$x_{rsv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R \cup \{0\}, \forall v \in V, \tag{6.5}$$

$$x_{rsv} \in \{0, 1\}, \quad \forall r \in R \cup \{0\}, \forall s \in R \cup \{|R| + 1\}, \forall v \in V. \tag{6.6}$$

The objective function (6.1) seeks to maximize the total profit from meeting the requests, and returns

the same optimal value on the objective function of formulation (2.1)-(2.5) from Section 2.3. Constraints (6.2) enforce that all vehicles depart from and arrive at the artificial depots as well as ensure the flow conservation of the vehicles on the request nodes. Constraints (6.3) enforce the number of vehicles that can be used to serve a given request. Constraints (6.4) and (6.5) prohibit the movements out of and into r, respectively, when vehicle type v cannot cross the arc where the demand is placed. Finally, constraints (6.6) impose the domain of decision variables. Note that if it is not possible to go from a node A to another node B in the space-time network, the value of the shortest path between these two nodes is ∞ and, hence, the profit between the requests related to these nodes becomes $-\infty$. We overcome the inconveniences resulting from setting Big-M profits by fixing $x_{rsv} = 0$ whenever $P_{rsv} = -\infty$.

The number of variables and constraints in the proposed model is equal to $|V||R|(|R|-1)/2$ and $|V||R|+|R|$, respectively (without considering the variables that are zeroed out). On the other hand, the number of variables and constraints in formulation (2.1)-(2.5) is equal to $|V||T||N|^2$ and $|V||T||N| + |T||N|^2$, respectively. As can be observed, the proposed model does not depend explicitly on the number of periods $|T|$ and terminals $|N|$ on the network, which can be of great advantage for solving some realistic large-scale instances with a moderate quantity of requests $|R|$ and vehicles $|V|$.

Finally, it is worth mentioning that formulation (6.1)–(6.6) has some similarities with a specialized formulation of the Full Truckload Vehicle Routing Problem (FTVRP) (Arunapuram et al., 2003; Desrosiers et al., 1984, 1988; Gronalt et al., 2003). However, there is a structural difference that does not enable us to use the solution methods previously developed for the FTVRP in an efficient manner for the VAP. In the FTVRP, it is necessary to serve (visit) all demand for shipping transport (customers), hence feasible and optimal solutions are Hamiltonian paths over directed graphs. Since in the VAP we consider the possibility of partially serving demand, solutions are simple paths in the time-space network. Furthermore, in the alternative formulation proposed in this paper, we are able to show that the graph is acyclic, thus, we can explore this property by using efficient algorithms in our decomposition approach. Finally, to the best of our knowledge, there is no other solution approach using a node-demand network for the VAP.

6.2 Dantzig-Wolfe decomposition

In this section, the Dantzig-Wolfe (DW) decomposition is presented for the new formulation of the VAP. We chose to decompose according to each vehicle since, from a practical point of view, it is best for operations control to create set of decisions according to vehicle groups or individual vehicles, and from a modelling perspective, quality results from Section 5.6 support this decision. Thus, by taking the demand-satisfying constraints (6.5) as the coupling constraints, the remaining constraints can be

grouped into the sets of solutions $X_v, \forall v \in V$, given by:

$$X_v = \left\{ x_v \quad | \quad \sum_{\substack{s \in R \cup \{|R|+1\} \\ s>r}} x_{rsv} - \sum_{\substack{s \in R \cup \{0\} \\ s<r}} x_{srv} = \left\{ \begin{array}{ll} 1 & \text{if } r = 0 \\ -1 & \text{if } r = |R|+1 \\ 0 & \text{if } r \neq 0 \wedge r \neq |R|+1 \end{array} \right\}, \right.$$
$$\forall r \in R \cup \{0\} \cup \{|R|+1\},$$
$$x_{srv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \qquad \forall r \in R \cup \{|R|+1\},$$
$$x_{rsv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \qquad \forall r \in R \cup \{0\},$$
$$\left. x_{rsv} \in \{0,1\}, \qquad \forall r \in R \cup \{0\}, s \in R \cup \{|R|+1\}. \right\}$$

Hence, the resulting equivalent formulation to (6.1)–(6.6) writes as:

$$\begin{aligned} \text{max} \quad & \sum_{v \in V} \sum_{r \in R \cup \{0\}} \sum_{s \in S \cup \{|R|+1\}} P_{rsv} x_{rsv} \\ \text{s.t.} \quad & \sum_{v \in V'} \sum_{\substack{s \in R \\ s>r}} x_{rsv} \leq D_r, \quad \forall r \in R, \\ & x_v \in X_v, \quad \forall v \in V, \end{aligned}$$

Each set X_v is a bounded polyhedron as it is defined exclusively by binary variables. Then, using the discretization approach (Lübbecke and Desrosiers, 2005; Vanderbeck, 2005), we can write any solution $x_v \in X_v$ as an integer combination of the extreme points of X_v as follows:

$$x_v = \sum_{q \in Q_v} \lambda_{vq} \bar{x}_{vq}, \text{ with } \sum_{q \in Q_v} \lambda_{vq} = 1, \lambda_{vq} \in \{0,1\}, \tag{6.7}$$

where $\bar{x}_{vq}$ denotes the extreme points of X_v, and Q_v is the set of indices of all extreme points. By substituting this representation into (6.1)-(6.6), the result is the Master Problem (MP):

$$\text{max} \quad \sum_{v \in V} \sum_{r \in R \cup \{0\}} \sum_{s \in S \cup \{|R|+1\}} P_{rsv} \left(\sum_{q \in Q_v} \lambda_{vq} \bar{x}_{vq} \right) \tag{6.8}$$

$$\text{s.t.} \quad \sum_{v \in V'} \sum_{\substack{s \in R \\ s>r}} \left(\sum_{q \in Q_v} \lambda_{vq} \bar{x}_{vq} \right) \leq D_r, \quad \forall r \in R, \qquad (u_r) \tag{6.9}$$

$$\sum_{q \in Q_v} \lambda_{vq} = 1, \qquad \forall v \in V, \qquad (w_v) \tag{6.10}$$

$$\lambda_{vq} \in \{0,1\}, \qquad \forall v \in V, \forall q \in Q_v, \tag{6.11}$$

where u_r and w_v represent the dual variables regarding the coupling and convexity constraints, re-

spectively, for the linear programming (LP) relaxation of the MP. Given the huge size of Q_v in realistic problem instances and the fact that most variables are likely to assume the value of zero in an optimal solution, we solve the LP relaxation of the MP using the Column Generation (CG) technique. More specifically, we initialize the LP relaxation of the MP with just a subset of extreme points of X_v (in our implementation, we use the empty path from node 0 to node $|R|+1$ in the request network, for each vehicle v), resulting in what is called the Restricted Master Problem (RMP). Hence, the RMP starts with a relatively small number of columns (variables) and we may generate new columns iteratively using the dual solutions and the following pricing subproblems:

$$Z_{sp(v)} = \max \sum_{r \in R \cup \{0\}} \sum_{s \in S \cup \{|R|+1\}} (P_{rsv} - u_r) x_{rsv} \tag{6.12}$$

$$\text{s.t.} \sum_{\substack{s \in R \cup \{|R|+1\} \\ s > r}} x_{rsv} - \sum_{\substack{s \in R \cup \{0\} \\ s < r}} x_{srv} = \begin{cases} 1 & \text{if } r = 0 \\ -1 & \text{if } r = |R|+1 \\ 0 & \text{if } r \neq 0 \wedge r \neq |R|+1 \end{cases}$$

$$\forall r \in R \cup \{0\} \cup \{|R|+1\} \tag{6.13}$$

$$x_{srv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R \cup \{|R|+1\}, \tag{6.14}$$

$$x_{rsv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R \cup \{0\}, \tag{6.15}$$

$$x_{rsv} \in \mathbb{R}_+, \quad \forall r \in R \cup \{0\}, s \in R \cup \{|R|+1\}, \tag{6.16}$$

for each $v \in V$. Hence, the reduced cost of a variable λ_{vq} is given by $Z_{sp(v)} - w_v$. In this particular case of the addressed problem, the optimal solution obtained at the end of the CG procedure has the same objective value as an optimal solution of the linear programming (LP) relaxation of the original model (6.1)–(6.6), as the pricing subproblems (6.12)-(6.16) have the integrality property. Nevertheless, the reformulation is still computationally attractive as the size of the MP is significantly reduced in comparison to the size of model (6.1)–(6.6) and the pricing subproblems can be efficiently solved using specialized shortest path algorithms on Directed Acyclic Graphs (DAG), as discussed in the next section.

6.3 Branch-and-price method

In this section, we propose a BP method to solve the MP (6.8)–(6.11), which consists in using the CG technique within each node of Branch-and-Bound (BB) tree (Lübbecke and Desrosiers, 2005). Our implementation is based on efficient approaches to enhance the overall performance of the method, such as a stabilized interior-point column generation technique (Subsection 6.3.1); a specialized algorithm for solving the pricing subproblems (Subsection 6.3.2); and a hierarchical branching strategy that impose constraints in the master problem (Subsection 6.3.3).

6.3.1 Interior-point column generation technique

At each node of the BP tree, we need to solve the LP relaxation of the MP (6.8)–(6.11) with possibly additional branching constraints. To avoid the well-known pathological behaviors resulting from using extreme dual solutions provided by the simplex method when solving the RMP (Vanderbeck, 2005), and in accordance with the promising results obtained in Chapter 5, we use the Primal-Dual Column Generation Method (PDCGM) (Gondzio et al., 2013, 2016) for solving the LP relaxations. The PDCGM is a stabilized column generation technique that relies on a primal-dual interior point method to solve each RMP. Hence, the obtained dual solutions are well-centered points in the dual feasible set and promote a better overall performance of the method, particularly for earlier CG approaches developed for the VAP (Cruz et al., 2019, 2020) and related problems (Alvarez and Munari, 2017; Gondzio and Munari, 2015; Munari and Gondzio, 2013).

After the CG method finishes, we check if the optimal solution provided is fractional and, if it is, we run a simple model-based heuristic in an attempt to quickly obtain an improved incumbent solution. This heuristic consists of (1) imposing integrality on the variables in the RMP solved in the last iteration of the CG method; and (2) solving the resulting integer programming problem by a general-purpose solver using a short time limit.

6.3.2 Pricing subproblem

Each subproblem (6.12)-(6.16) is a maximum cost flow problem over a DAG in which we have to flow one vehicle from the starting depot (node 0) to the end depot (node $|R| + 1$), hence it is a longest path problem (as we are maximizing profits) over a directed acyclic network. Since the graph for each vehicle is already topologically sorted, we can use a linear update for optimality conditions as described in Algorithm 4. The values $y_{r,s}$ correspond to the updated arc profits through the dual values from the coupling constraints of the master problem and whenever $A_{i_r j_r v} = 0$, we set $y_{r,s} = -\inf$.

Algorithm 4 DAG's algorithm

procedure DAG($0, |R| + 1$)
  Initialize $d[k] = -\infty$ and $p[k] = 0, \forall k \in R \cup \{|R| + 1\}$;
  $d[0] = 0$;
  for $r \leftarrow 0 : R$ **do**
    for $s \leftarrow r + 1 : |R| + 1$ **do**
      let $y_{rs} = P_{rsv} - u_r$ be the profit of arc (r, s) for vehicle $v \in V$
      if $d[s] > d[r] + y_{rs}$ **then**
        $d[s] \leftarrow d[r] + y_{rs}$
        $p[s] \leftarrow r$
      end if
    end for
  end for
end procedure

6.3.3 Branching strategy

The branching strategy used in the proposed BP method consists of a hierarchical rule based on the total served demand and on the arc flow in a fractional solution. More specifically, given an optimal solution $\bar{\lambda}$ of the RMP solved in the last iteration of the CG technique, we verify if the total demand served by all vehicles for a given $r \in R$ is fractional, i.e.

$$\sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} \sum_{q \in Q_v} \bar{x}_{qrsv} \bar{\lambda}_{vq} \notin \mathbb{Z}_+. \tag{6.17}$$

Among all indices that satisfy (6.17), we select the index r with the earliest departing time and then generate two child nodes, one with the first and the other with the second of the following additional branching constraints:

$$\sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} \sum_{q \in Q_v} \bar{x}_{qrsv} \lambda_{vq} \geq \left\lceil \sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} \sum_{q \in Q_v} \bar{x}_{qrsv} \bar{\lambda}_{vq} \right\rceil, \tag{6.18}$$

$$\sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} \sum_{q \in Q_v} \bar{x}_{qrsv} \lambda_{vq} \leq \left\lfloor \sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} \sum_{q \in Q_v} \bar{x}_{qrsv} \bar{\lambda}_{vq} \right\rfloor, \tag{6.19}$$

where the unary operators $\lceil\ \rceil$ and $\lfloor\ \rfloor$ represent the ceiling and floor functions for rounding fractional numbers. Since they are imposed in the LP relaxation of the MP, these branching constraints do not damage the structure of the pricing subproblems. To account for the dual solution in the pricing subproblems in a given node k, we only need to include the following summation in the objective function of vth subproblem, for each $v \in V$:

$$- \sum_{r \in B^k} \sum_{s \in S \cup \{|R|+1\}} \bar{b}_r x_{rsv},$$

where B^k is the set of all indices $r \in R$ such that a branching constraint of type (6.18) or (6.19) is imposed in node k (coming from all its ancestral nodes in the tree), and $\bar{b}_r$ is the dual solution associated to the rth branching constraint of these types.

This branching rule alone does not guarantee an integer solution as the summation in (6.17) may be integer even for a solution $\bar{\lambda}$ with fractional components. Hence, we resort to an additional rule when the summation in (6.17) is integer for all $r \in R$. This rule is based on the values of variables x_{rsv}, which are computed from solution $\bar{\lambda}$ using the summation in (6.7). Requiring integrality of x_{rsv} is equivalent to requiring integrality of the MP variables λ, as we rely on discretization (Lübbecke and Desrosiers, 2005). Hence, given a tuple of indices (r, s, v) such that x_{rsv} is fractional, we branch by

enforcing $x_{rsv} = 0$ in one child node, and $x_{rsv} = 1$ in the other child node. The selection of (r, s, v) is done giving priority to variables with earlier departing times, choosing randomly among different vehicles to break ties. To avoid damaging the structure of the subproblems, we impose the new bounds of x_{rsv} at the MP level, by inserting one of the following constraints:

$$\sum_{q \in Q^v} \bar{x}_{qrsv} \lambda_{vq} \leq 0 \text{ or } \sum_{q \in Q^v} \bar{x}_{qrsv} \lambda_{vq} \geq 1, \tag{6.20}$$

which correspond to imposing $x_{rsv} \leq 0$ and $x_{rsv} \geq 1$, respectively. Then, similarly to the previous branching rule, we only need to modify the objective function of the pricing subproblems to account for the dual solution related to these new branching constraints. More specifically, in a given node k of the tree, we include the following summation in the objective function of the vth subproblem:

$$- \sum_{(r,s,v) \in U^k \cup L^k} w_{rsv} x_{rsv}$$

where w_{rsv} is the dual solution associated to constraints of type (6.20), and U^k and L^k are the set of index tuples (r, s, v) related to the upper and lower bound constraints in (6.20) that are imposed in node k. Note that the values w_{rsv} can be easily incorporated into the subproblems as additional costs at the arcs of the request network.

In summary, we have a hierarchical branching scheme in which we first verify if there is at least one index r that satisfy (6.17) and, if no such index exists, we verify if there is at least one tuple (r, s, v) such that x_{rsv} is fractional. Finally, it is worth mentioning that we use the best-first search rule to decide the next node to be processed along the search tree, i.e, the method selects the node with the worst dual bound (maximum dual bound in maximization problems).

6.4 Computational Experiments

In this section, we present the results of computational experiments with the proposed approaches as well as the approaches from Chapter 5 for solving the VAP, using small and large-scale realistic-sized instances collected from Vasco and Morabito (2016b) and Chapter 5. First, we present the results of solving the two considered compact formulations, namely the node-demand formulation proposed in Section 6.1 and the arc-demand formulation presented in Section 2.3. Then, we show the results of the experiments with the BP method proposed in Section 6.3 and the BP method proposed in Section 5. All methods were implemented in C++ and use the Concert Library of the IBM CPLEX Optimization Studio v.12.8.1. Additionally, for the BP method we use the PDCGM library (Gondzio et al., 2016) as the CG solver. All experiments reported in this section were run on a PC with CPU Intel Core i7-4790S

3.20GHz and 16 GB of RAM. A symbol "*" indicates that CPLEX could not solve or even mount the model due to lack of computer memory. A time limit of 7200 seconds was imposed on each run.

We use two set of instances to develop the performance analysis of the above-mentioned solution methods. The first set comprises 30 instances from the work of Vasco and Morabito (2016b) which share the same characteristics, namely, 53 terminals, 36 periods, 300 requests and 130 vehicles. The second set comprises the 200 generated instances from Chapter 5 with different number of terminals, periods, requests and vehicles. These instances will be referred to according to the range of the number of terminals: $[10, 19]$, $[20, 29]$, $[30, 39]$, $[40, 49]$ and $[50, 59]$. For each of these terminal ranges, the number of vehicles varies in the following way: for $[10, 19]$ it increases from 20 to 50 vehicles; for $[20, 29]$, from 100 to 150; for $[30, 39]$, from 200 to 250; for $[40, 49]$, from 130 to 170; and for $[50, 59]$, from 100 to 250. The numbers of periods are 10, 30, 36 and 36, while the number of requests are 50, 200, 300, 500 and 700 for each terminal range, respectively. The detailed results of solving these instances with the methods are presented in the Appendix C.

6.4.1 Comparison of arc-demand formulation and node-demand formulation

In this section, we compare the performance of the general-purpose ILP solver of CPLEX in solving the compact formulations of Sections 2.3 and 6.1. Table 6.1 shows the computational times for solving the LP relaxation and the ILP model of the arc-demand formulation (2.1)–(2.5) and the node-demand formulation (6.1)–(6.6) for the set of instances [20,29], [30,39] of Chapter 5 and the 30 large-scale instances of Vasco and Morabito (2016b). Table 6.1 includes only the computational times of the LP relaxation and the corresponding ILP model for both formulations of instances [20,29], [30,39] of Chapter 5. Computational times for the arc-demand formulation of instances from Vasco and Morabito (2016b) are not included as CPLEX could not mount the corresponding models. It is worth mentioning that the respective ILP arc-demand models of these instances have 26,292,240 variables and 349,164 constraints. Objective values for all instances are detailed in Appendix C. Results for instances [10,19] are omitted as all computational times do not exceed 1 second. In the case of instances [40,49], CPLEX only solved 2 and 11 instances to optimality for the arc-demand and node-demand formulations, respectively, which are also detailed in Appendix C.

From Table 6.1, we observe that CPLEX in solving the model of the node-demand formulation outperformed that of the arc-demand formulation as it solved faster all instances in the sets [20,29] and [30,39]. The solution of the LP relaxation of arc-demand model required up to 20 and 269 seconds for the $[20, 29]$ and $[30, 39]$ instances, respectively, while that of the node-demand model required up to 3 and 19 seconds for the $[20, 29]$ and $[30, 39]$ instances, respectively. In the case of the ILP model, the difference in computational times is more pronounced. It took up to 58 and 934 seconds for the arc-

Table 6.1: Computational times for solving instances from Vasco and Morabito (2016b) and instances with terminals in ranges $[20, 29]$ and $[30, 39]$ using the arc-demand and node-demand formulations.

Instance	*Arc-demand*		*Node-demand*		*Instance*	*Arc-demand*		*Node-demand*		*Instance*	*Node-demand*	
	LP	*ILP*	*LP*	*ILP*		*LP*	*ILP*	*LP*	*ILP*		*LP*	*ILP*
20-20-100-200	3.49	16.23	1.56	2.64	30-30-200-300	100.44	392.92	7.26	9.54	53-36-130-300-p1	4.15	28.81
20-20-130-200	4.33	19.13	1.26	2.26	30-30-230-300	104.13	435.42	6.84	9.18	53-36-130-300-p2	3.19	5.05
20-20-150-200	5.45	22.10	1.73	2.97	30-30-250-300	153.42	750.87	13.72	16.14	53-36-130-300-p3	2.34	4.84
21-20-100-200	5.36	19.09	1.58	2.48	31-30-200-300	98.37	436.45	7.82	10.46	53-36-130-300-p4	4.55	29.11
21-20-130-200	5.01	21.65	1.30	2.27	31-30-230-300	135.00	716.02	10.75	13.15	53-36-130-300-p5	2.01	2.75
21-20-150-200	5.51	23.34	1.75	2.96	31-30-250-300	139.71	685.50	12.64	14.95	53-36-130-300-p6	5.79	12.30
22-20-100-200	5.82	23.76	1.46	2.56	32-30-200-300	132.54	497.74	7.67	9.83	53-36-130-300-p7	2.81	3.42
22-20-130-200	5.71	25.56	1.88	2.89	32-30-230-300	130.23	621.84	9.94	11.45	53-36-130-300-p8	4.01	14.68
22-20-150-200	7.14	34.33	2.68	4.20	32-30-250-300	154.61	730.32	14.49	16.02	53-36-130-300-p9	3.51	3.85
23-20-100-200	4.65	19.48	1.11	1.84	33-30-200-300	110.64	456.08	9.10	10.48	53-36-130-300-p10	3.75	2.64
23-20-130-200	6.44	27.30	1.62	2.90	33-30-230-300	268.12	933.43	12.97	14.46	53-36-130-300-p11	2.09	3.39
23-20-150-200	8.00	31.67	2.15	3.53	33-30-250-300	101.29	417.62	7.85	9.35	53-36-130-300-p12	4.50	15.54
24-20-100-200	5.30	24.04	1.33	2.29	34-30-200-300	131.44	639.94	10.56	11.93	53-36-130-300-p13	3.37	6.87
24-20-130-200	11.08	42.33	2.56	3.51	34-30-230-300	164.23	903.13	14.56	16.45	53-36-130-300-p14	2.90	4.34
24-20-150-200	12.68	52.91	2.38	3.87	34-30-250-300	*	*	12.80	14.17	53-36-130-300-p15	4.04	4.48
25-20-100-200	5.20	23.05	1.35	2.53	35-30-200-300	115.12	536.07	7.17	9.46	53-36-130-300-p16	3.30	5.05
25-20-130-200	8.86	38.92	2.30	4.29	35-30-230-300	*	*	10.95	12.96	53-36-130-300-p17	2.20	2.24
25-20-150-200	9.41	37.72	2.08	2.97	35-30-250-300	*	*	10.39	12.87	53-36-130-300-p18	3.22	5.99
26-20-100-200	5.49	22.13	1.37	2.50	36-30-200-300	116.12	615.03	8.72	11.07	53-36-130-300-p19	2.00	2.23
26-20-130-200	7.66	33.17	1.59	2.93	36-30-230-300	*	*	12.44	15.01	53-36-130-300-p20	2.78	5.51
26-20-150-200	19.33	67.78	2.39	4.36	36-30-250-300	*	*	18.74	20.48	53-36-130-300-p21	7.36	16.13
27-20-100-200	7.05	32.24	1.66	2.38	37-30-200-300	144.08	826.57	9.75	11.46	53-36-130-300-p22	5.56	16.86
27-20-130-200	11.71	55.14	2.34	3.48	37-30-230-300	*	*	9.46	11.01	53-36-130-300-p23	4.41	13.97
27-20-150-200	9.95	44.40	1.81	2.89	37-30-250-300	*	*	10.59	12.85	53-36-130-300-p24	2.31	2.60
28-20-100-200	10.32	42.01	1.89	2.89	38-30-200-300	*	*	7.21	9.07	53-36-130-300-p25	3.43	6.48
28-20-130-200	8.64	36.32	1.44	2.65	38-30-230-300	*	*	11.67	13.96	53-36-130-300-p26	2.63	3.50
28-20-150-200	11.36	57.47	2.41	4.07	38-30-250-300	*	*	12.50	14.76	53-36-130-300-p27	2.67	2.69
29-20-100-200	7.09	31.58	1.28	2.15	39-30-200-300	*	*	8.57	11.01	53-36-130-300-p28	2.79	3.82
29-20-130-200	11.48	49.86	1.88	2.77	39-30-230-300	*	*	13.52	16.77	53-36-130-300-p29	2.21	2.29
29-20-150-200	10.10	44.10	1.88	3.22	39-30-250-300	*	*	10.49	12.32	53-36-130-300-p30	2.69	3.63

demand model in both set of instances, while it took up to 5 and 21 seconds for node-demand model across both set of instances. Note that CPLEX could not mount the models of 13 instances in the set $[30, 39]$. In the case of the larger instances from Vasco and Morabito (2016b), by using the proposed node-demand model CPLEX was able to solve the LP relaxation of all instances in less than 8 seconds and the ILP models in less than 30 seconds. It is worth mentioning, however, that many of the 300 requests describing these instances overlap on the same subset of arcs; hence, all of the instances have a number of demand arcs in the range 50 to 70.

Table 6.2 presents the average computational times of solving the LP ($Av\,LP$) and ILP ($Av\,ILP$)

models, and the number of instances solved (*Count*) for the arc-demand and node-demand formulations. Instances in the range [50,59] are not included as neither the arc-demand nor the node-demand models could be mounted by CPLEX. From this table we observe that the computational times resulting from using the node-demand formulation are better in each set of instances. Additionally, for the set [30,39], [40,49] and the ones from Vasco and Morabito (2016b), CPLEX was able to solve a larger proportion of instances using the proposed node-demand formulation.

Table 6.2: Average computational times and count of instances solved to optimality by CPLEX using the arc-demand and node-demand formulations.

Instance	*Arc-demand*			*Node-demand*		
	Av LP	*Av ILP*	*Count*	*Av LP*	*Av ILP*	*Count*
[10,19]	0.12	0.31	50	0.01	0.04	50
[20,29]	7.99	33.96	30	1.80	2.97	30
[30,39]	135.26	623.23	17	10.70	12.76	30
[40,49]	321.51	1891.44	2	24.44	27.98	11
Vasco (2016)	*	*	0	3.44	7.82	30

6.4.2 Comparison of the BP methods based on the reformulations the arc-demand and node-demand formulations

In this section, we analyze the performance of the BP method proposed in Section 6.3, based on the reformulation of the node-demand model. Tables 6.3, 6.4 and 6.5 show the results of the BP method proposed in this chapter and the BP method from Chapter 5, which is based on the reformulation of the arc-demand model. Columns in all these tables refer to: T_{root} is the time taken to solve the root node only and T_{BP} is the time taken by the BP. The detailed results for all sets of instances are shown in Appendix C.

Table 6.3 shows the results of both BP methods in instances from the work of Vasco and Morabito (2016b). We observe that even though both BP were able to process and solve the root node of all instances, the node-demand BP method clearly outperforms the arc-demand BP method in terms of computational times. All instances were solved in less than 45 seconds by the arc-based BP method, while it took less than 1 second for the BP method based on the node-demand model. The six instances with boldfaced names are the only ones within this set that have positive integrality gaps at the root node, i.e., the optimal value obtained at the root node is strictly larger than the optimal value of the instance. For these instances, the node-demand BP method took less than 3 seconds, while the arc-demand BP method took more than 46 seconds.

Table 6.4 shows the results of the BP methods in instances with terminals in ranges $[20, 29]$ and

Table 6.3: Results of the BP methods based on the arc-demand and node-demand formulations for instances from Vasco and Morabito (2016b).

Instance	*Arc-demand*		*Node-demand*		*Instance*	*Arc-demand*		*Node-demand*	
	T_{Root}	T_{BP}	T_{Root}	T_{BP}		T_{Root}	T_{BP}	T_{Root}	T_{BP}
53-36-130-300-p1	33.0	46.9	0.5	0.5	53-36-130-300-p16	36.0	36.0	0.2	0.2
53-36-130-300-p2	29.1	29.1	0.2	0.2	53-36-130-300-p17	33.5	33.5	0.2	0.2
53-36-130-300-p3	31.0	409.1	0.4	1.2	**53-36-130-300-p18**	36.6	83.0	0.5	0.8
53-36-130-300-p4	36.8	36.8	0.3	0.3	53-36-130-300-p19	37.2	37.2	0.2	0.2
53-36-130-300-p5	29.1	29.1	0.2	0.2	53-36-130-300-p20	36.3	36.3	0.2	0.2
53-36-130-300-p6	33.0	33.0	0.3	0.3	53-36-130-300-p21	44.8	44.8	0.3	0.3
53-36-130-300-p7	31.5	31.5	0.3	0.3	53-36-130-300-p22	37.6	37.6	0.4	0.4
53-36-130-300-p8	29.2	127.9	0.2	1.5	53-36-130-300-p23	34.4	34.4	0.3	0.3
53-36-130-300-p9	36.7	36.7	0.3	0.3	53-36-130-300-p24	27.3	27.3	0.2	0.2
53-36-130-300-p10	25.2	25.2	0.2	0.2	53-36-130-300-p25	33.9	33.9	0.3	0.3
53-36-130-300-p11	36.0	46.5	0.2	2.0	53-36-130-300-p26	30.2	30.2	0.2	0.2
53-36-130-300-p12	34.7	34.7	0.3	0.3	53-36-130-300-p27	27.1	27.1	0.2	0.2
53-36-130-300-p13	37.5	80.6	0.1	0.9	53-36-130-300-p28	37.9	37.9	0.2	0.2
53-36-130-300-p14	31.8	31.8	0.2	0.2	53-36-130-300-p29	30.2	30.2	0.2	0.2
53-36-130-300-p15	38.6	38.6	0.2	0.2	53-36-130-300-p30	33.7	33.7	0.2	0.2

$[30, 39]$. First, we compare the performance of the methods in solving the root node. In the set $[20, 29]$, the node-demand BP method took less than 1 second, while the arc-demand BP method took less than 4 seconds. In the set $[30, 39]$, the difference is slightly more pronounced as it still took less than 1 second for the node-demand BP method and more than 9 seconds for the arc-demand BP method for all instances. Regarding the performance of the BP methods to solve the instances to proven optimality, the node-demand BP method solved all instances in the set $[20, 29]$ in less than 37 seconds, whereas the arc-demand BP method took 73 seconds (both maximum computational times occurred in instance 29-20-130-200). In the set $[30, 39]$, it took less than 383 seconds for the node-demand BP method and 1540 seconds for the arc-demand BP method (both maximum computational times occurred in instance 31-30-230-300). It is worth highlighting that in all instances in this table, the proposed BP method was faster than the arc-demand BP method of Chapter 5.

Finally, Table 6.5 shows the results of the BP methods in instances with terminals in ranges $[40, 49]$ and $[50, 59]$. Instances with boldfaced times (BP) correspond to instances that were not solved to proven optimality within the time limit by the BP method of Chapter 5. These instances are 50-36-130-700 and 50-36-250-700, with upper bounds (lower bounds) of 57051.5 (57048.0) and 111448.0 (111444.0), respectively; and both with final relative gaps of 0.0001%. Apart from these two, all remaining instances

Table 6.4: Results of the BP methods based on the arc-demand and node-demand formulations for instances with terminals in ranges $[20, 29]$ and $[30, 39]$.

Instance	*Arc-demand*		*Node-demand*		*Instance*	*Arc-demand*		*Node-demand*	
	T_{Root}	T_{BP}	T_{Root}	T_{BP}		T_{Root}	T_{BP}	T_{Root}	T_{BP}
20-20-100-200	1.48	5.36	0.09	1.15	30-30-200-300	8.23	16.41	0.36	3.96
20-20-130-200	1.51	7.51	0.11	4.26	30-30-230-300	9.41	21.33	0.46	6.41
20-20-150-200	1.70	2.63	0.12	1.10	30-30-250-300	10.03	108.11	0.51	71.58
21-20-100-200	1.75	3.09	0.08	0.98	31-30-200-300	8.46	11.79	0.30	2.08
21-20-130-200	2.05	3.51	0.13	1.02	31-30-230-300	9.68	1539.31	0.41	382.25
21-20-150-200	1.88	3.56	0.12	1.06	31-30-250-300	10.34	13.13	0.49	4.01
22-20-100-200	1.96	5.71	0.10	1.36	32-30-200-300	9.11	39.08	0.35	9.51
22-20-130-200	1.99	4.04	0.11	0.94	32-30-230-300	10.83	18.04	0.40	2.51
22-20-150-200	2.12	5.47	0.12	1.40	32-30-250-300	17.02	394.39	0.57	176.37
23-20-100-200	1.90	2.99	0.09	0.75	33-30-200-300	9.86	254.35	0.35	21.98
23-20-130-200	1.94	3.02	0.09	1.17	33-30-230-300	11.13	32.99	0.40	6.21
23-20-150-200	2.20	5.03	0.12	1.93	33-30-250-300	11.59	65.80	0.41	16.85
24-20-100-200	2.37	6.28	0.08	1.32	34-30-200-300	14.44	63.03	0.45	14.86
24-20-130-200	3.51	13.02	0.14	5.79	34-30-230-300	12.19	56.14	0.45	11.04
24-20-150-200	3.26	9.43	0.15	2.36	34-30-250-300	12.77	21.54	0.44	4.00
25-20-100-200	2.33	5.93	0.07	1.09	35-30-200-300	11.06	170.60	0.34	34.44
25-20-130-200	2.64	4.07	0.12	0.99	35-30-230-300	13.41	45.84	0.40	6.39
25-20-150-200	2.91	6.05	0.14	1.80	35-30-250-300	13.90	61.65	0.44	13.50
26-20-100-200	2.64	4.16	0.08	0.73	36-30-200-300	13.64	29.92	0.39	4.82
26-20-130-200	2.76	3.93	0.10	0.70	36-30-230-300	14.57	28.51	0.44	6.79
26-20-150-200	3.94	21.92	0.14	5.35	36-30-250-300	14.50	21.88	0.47	2.76
27-20-100-200	3.31	4.54	0.08	0.76	37-30-200-300	13.07	16.96	0.29	6.33
27-20-130-200	3.95	29.05	0.14	6.51	37-30-230-300	16.65	1250.38	0.39	273.13
27-20-150-200	3.29	6.69	0.12	1.94	37-30-250-300	16.07	48.15	0.41	7.76
28-20-100-200	3.23	42.24	0.08	8.75	38-30-200-300	14.08	465.01	0.34	75.33
28-20-130-200	3.32	4.63	0.10	1.19	38-30-230-300	15.72	25.18	0.44	3.72
28-20-150-200	3.73	5.49	0.12	1.15	38-30-250-300	18.31	26.90	0.48	4.05
29-20-100-200	3.69	7.48	0.08	1.56	39-30-200-300	18.71	69.13	0.39	9.22
29-20-130-200	3.60	72.53	0.12	36.02	39-30-230-300	17.87	238.48	0.43	44.26
29-20-150-200	3.91	6.73	0.11	0.76	39-30-250-300	17.57	60.95	0.46	9.59

Table 6.5: Results of the BP methods based on the arc-demand and node-based formulations for instances with terminals in ranges $[40, 49]$ and $[50, 59]$.

Instance	*Arc-demand*		*Node-demand*		Instance	*Arc-demand*		*Node-demand*		Instance	*Arc-demand*		*Node-demand*	
	T_{Root}	T_{BP}	T_{Root}	T_{BP}		T_{Root}	T_{BP}	T_{Root}	T_{BP}		T_{Root}	T_{BP}	T_{Root}	T_{BP}
40-36-130-500	15.23	47.90	0.46	11.66	50-36-100-700	23.56	34.71	0.69	7.61	55-36-100-700	29.75	100.46	0.68	16.99
40-36-150-500	17.29	32.65	0.62	8.89	50-36-130-700	29.24	**7200.00**	1.02	5822.36	55-36-130-700	47.77	73.97	1.02	12.71
40-36-170-500	18.22	52.27	0.69	11.44	50-36-150-700	31.62	68.35	1.19	20.72	55-36-150-700	40.81	49.44	1.16	14.83
41-36-130-500	18.11	113.69	0.48	27.03	50-36-180-700	34.10	627.10	1.39	245.91	55-36-180-700	43.37	109.13	1.37	27.87
41-36-150-500	19.53	24.16	0.62	5.85	50-36-200-700	34.85	37.93	1.50	10.51	55-36-200-700	45.23	120.55	1.52	30.83
41-36-170-500	20.81	25.21	0.70	4.64	50-36-250-700	40.64	**7200.00**	1.91	761.11	55-36-250-700	55.72	108.08	2.15	34.86
42-36-130-500	19.18	42.48	0.47	6.05	51-36-100-700	24.03	99.31	0.59	25.22	56-36-100-700	35.63	54.07	0.69	5.23
42-36-150-500	22.40	26.89	0.78	7.63	51-36-130-700	30.62	42.11	1.02	9.77	56-36-130-700	52.44	57.63	1.03	13.00
42-36-170-500	21.62	80.60	0.67	62.20	51-36-150-700	30.51	171.96	1.16	56.61	56-36-150-700	41.97	98.02	1.17	20.91
43-36-130-500	20.24	25.25	0.46	4.94	51-36-180-700	37.18	1798.33	1.39	777.42	56-36-180-700	48.28	244.86	1.37	73.35
43-36-150-500	21.36	138.62	0.61	30.75	51-36-200-700	38.92	55.16	1.72	10.71	56-36-200-700	51.13	65.10	1.32	14.93
43-36-170-500	21.77	65.34	0.68	15.47	51-36-250-700	49.22	772.57	2.18	268.88	56-36-250-700	54.70	1942.71	2.12	546.20
44-36-130-500	20.28	79.92	0.47	16.84	52-36-100-700	25.76	38.97	0.68	7.48	57-36-100-700	33.38	49.92	0.68	7.60
44-36-150-500	22.88	39.81	0.61	7.70	52-36-130-700	33.24	63.88	1.02	11.91	57-36-130-700	37.13	153.09	0.87	30.21
44-36-170-500	25.29	128.38	0.68	26.54	52-36-150-700	28.20	46.27	0.89	7.81	57-36-150-700	40.73	80.27	1.00	13.53
45-36-130-500	21.41	38.40	0.46	7.64	52-36-180-700	37.01	208.32	1.22	73.72	57-36-180-700	47.38	62.52	1.54	13.76
45-36-150-500	22.82	69.02	0.53	13.67	52-36-200-700	43.32	91.11	1.53	18.60	57-36-200-700	52.76	57.93	1.72	10.89
45-36-170-500	25.23	35.68	0.60	6.51	52-36-250-700	52.17	224.81	2.19	9.48	57-36-250-700	62.00	63.98	2.12	19.03
46-36-130-500	26.07	393.58	0.61	69.61	53-36-100-700	26.39	150.27	0.58	58.73	58-36-100-700	48.58	99.66	1.19	27.76
46-36-150-500	24.76	232.55	0.61	62.85	53-36-130-700	31.03	51.06	0.88	12.69	58-36-130-700	31.85	49.34	0.58	9.71
46-36-170-500	28.27	37.83	0.67	6.55	53-36-150-700	38.67	55.33	1.15	11.14	58-36-150-700	53.58	51.38	1.50	11.63
47-36-130-500	34.85	69.29	0.69	11.31	53-36-180-700	38.71	1023.86	1.39	314.93	58-36-180-700	42.30	164.57	0.98	34.26
47-36-150-500	26.57	76.43	0.60	42.31	53-36-220-700	42.52	105.90	1.44	43.48	58-36-200-700	62.03	80.03	2.12	30.48
47-36-170-500	30.42	78.83	0.69	13.71	53-36-250-700	49.47	65.19	1.90	18.85	58-36-250-700	46.64	2271.38	1.14	632.19
48-36-130-500	28.28	91.27	0.59	15.70	54-36-100-700	29.48	137.10	0.68	31.67	59-36-100-700	46.65	511.11	0.78	99.67
48-36-150-500	33.14	118.50	0.72	20.66	54-36-130-700	33.05	45.46	1.01	9.91	59-36-130-700	50.34	1147.60	1.28	201.27
48-36-170-500	29.44	163.78	0.59	26.56	54-36-150-700	38.63	81.05	1.16	24.17	59-36-150-700	60.56	79.93	1.12	14.84
49-36-130-500	28.81	37.78	0.52	5.06	54-36-180-700	40.40	107.63	1.56	35.34	59-36-180-700	52.15	256.82	1.36	56.59
49-36-150-500	40.63	121.98	0.70	19.01	54-36-200-700	41.92	45.06	1.31	14.89	59-36-200-700	55.35	73.87	1.31	14.88
49-36-170-500	34.51	72.37	0.69	14.07	54-36-250-700	57.90	1418.64	2.14	469.17	59-36-250-700	63.43	82.01	1.86	24.60

reached optimality within the time limit. In the set $[40, 49]$, for solving the root node only, the node-based BP method took less than 1 second while the arc-demand BP method took more than 15 seconds, considering all instances. In the set $[50, 59]$, it took less than 3 seconds for the node-demand BP method while it took more than 23 seconds for the arc-demand BP method for all instances. Regarding the integer optimal solutions, the node-demand BP method clearly outperformed the arc-demand BP. In the set, $[40, 49]$ it took less 70 seconds for the method based on the node-demand model and less than 394 seconds for the method based on the arc-demand model. In the set $[50, 59]$, apart from the instances not solved by the arc-demand BP method, the node-demand BP method took less than 778 seconds, while the arc-demand BP method took less than 2272 seconds. Note that both of the above mentioned instances that could not be solved by the arc-demand BP method, were indeed solved by the node-demand BP method in 2594.82 and 761.11 seconds. As observed in the results presented in the previous tables, the proposed BP method was also faster than the BP method of Chapter 5 in all instances in the sets $[40, 49]$ and $[50, 59]$.

Table 6.6 presents the average computational times of solving the root node only ($Av\,T_{Root}$), the average computational time of solving the integer problem ($Av\,T_{BP}$) and the number of instances solved (*Count*) for the arc-demand and node-demand formulations. From this table we observe that the computational times resulting from using the node-demand formulation are better in each set of instances. Additionally, we were able to solve two unsolved instances with respect to Chapter 5.

Table 6.6: Average computational times and count of instances solved to optimality using the BP methods based on the arc-demand and node-demand formulations.

Instance	*Arc-demand*			*Node-demand*		
	$Av\,T_{Root}$	$Av\,T_{BP}$	*Count*	$Av\,T_{Root}$	$Av\,T_{BP}$	$Count$
[10,14]	0.12	0.38	15	0.01	0.09	15
[15,19]	0.21	0.54	15	0.01	0.04	15
[20,29]	2.70	10.20	30	0.11	3.20	30
[30,39]	13.14	173.83	30	0.42	41.19	30
[40,49]	24.65	85.35	30	0.61	19.43	30
[50,54]	36.41	496.03	28	1.28	198.75	30
[55,59]	47.79	278.65	30	1.29	68.82	30
(Vasco and Morabito, 2016b)	33.67	53.35	30	0.27	0.43	30

In summary, from the results shown in this section regarding the addressed compact models and BP methods, it can be observed that the proposed node-demand representation yields superior results in computational efficiency for realistic-sized instances using both the compact model in a general-purpose ILP solver (CPLEX) and the tailored exact solution method based on column generation. By

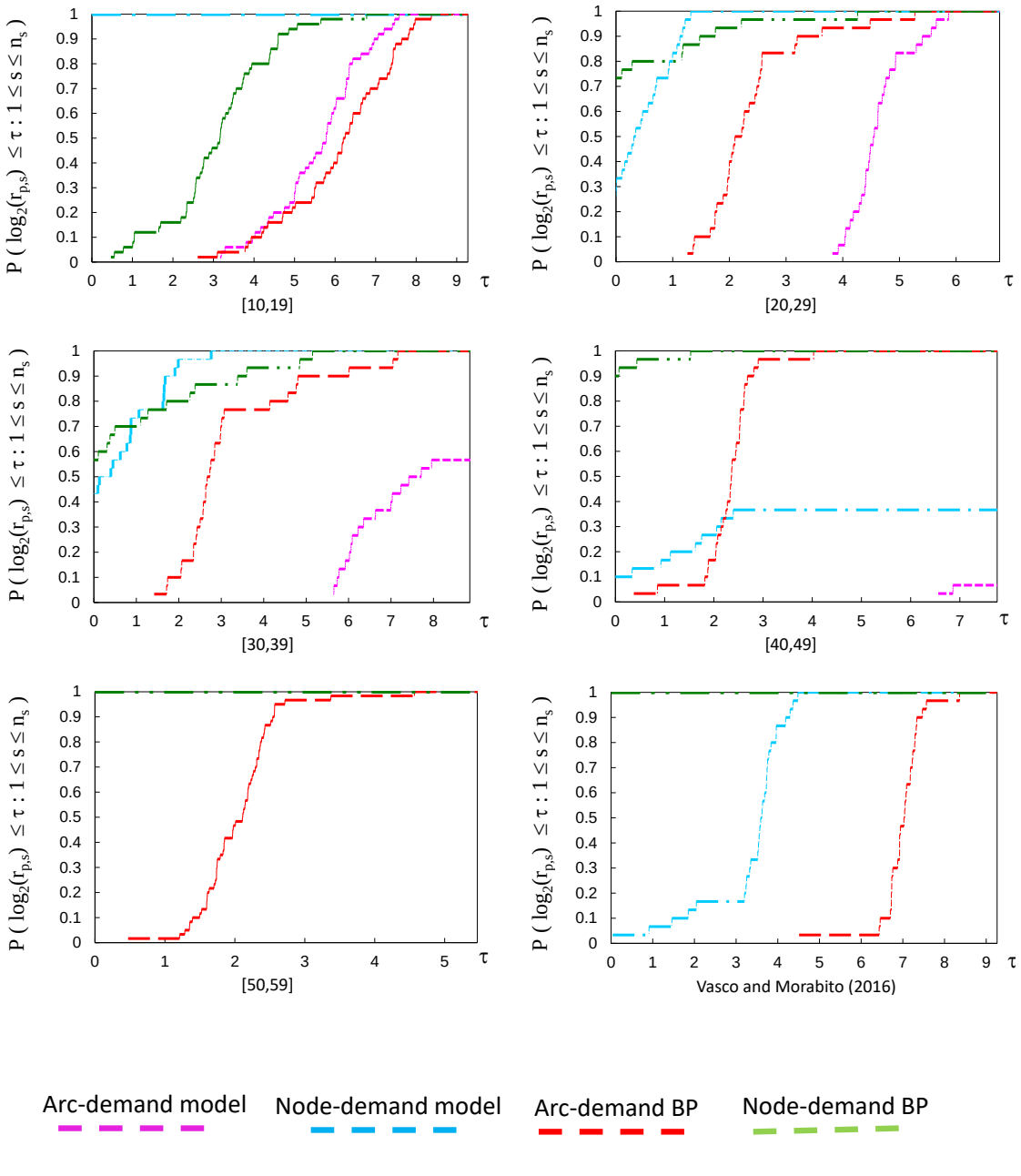
using CPLEX with the node-demand formulation, we were able to solve all instances of Vasco and Morabito (2016b), while these same instances could not be even mounted in the case of the arc-demand formulation. Furthermore, we were able to solve a larger proportion of instances from the 200 instances of Chapter 5 using CPLEX as well. Regarding the BP method, we obtained better results in both classes of instances, Vasco and Morabito (2016b) and Chapter 5, as we were able to solve all of them faster with the new formulation. In addition, we were able to solve to optimality the two instances that could not be solved by the BP method from Chapter 5.

Figure 6.6 shows the performance profiles (Dolan and Moré, 2002) based on computational times for each set of instances, considering the four solution approaches presented in this work, i.e., the two compact models solved by CPLEX and the two BP methods. The value $P(\tau)$ for a given method corresponds to the fraction of instances for which that method provides solutions with a computational time within a factor of 2^τ of the best computational time. When $\tau = 0$, the value $P(\tau)$ indicates the proportion of instances for which a given method performed the best, i.e., was the fastest; when $\tau \to \infty$, the $P(\tau)$ indicates the proportion of instances that were solved by a given method. In sets $[10, 19]$, $[20, 29]$ and $[30, 39]$, all four methods were able to solve all instances. The compact node-demand model is the most efficient in solving the set of smallest instances, i.e., $[10, 19]$. For sets $[20, 29]$ and $[30, 39]$, the node-based BP method is the fastest in most instances, although it takes longer than the corresponding compact model to prove optimality for a few instances in these sets. Then, for the sets $[40, 49]$ and $[50, 59]$, the node-demand BP method clearly superior to any other approach, as it is the fastest in more than 90% of instances and is the first to prove optimality for all instances.

6.5 Final Considerations

In this chapter, we have presented a new formulation and a tailored exact solution method based on the DW decomposition for solving the deterministic VAP. This new formulation consists in preprocessing the space-time network of the previous chapter, so that decisions can be reduced to exclusively allocating empty vehicles between requests. We have solved this new formulation using CPLEX. From the results obtained in this section, it can be seen that the new formulation poses greater advantages for solving the VAP. In addition, we used the DW reformulation and proposed a BP method which relies on solving the pricing problem using shortest path algorithms and a stabilized PDCGM for solving the master problem. Results from computational experiments show advantages in solving capabilities and time efficiency from using this tailored exact solution method compared to the method presented in the previous chapter.

Figure 6.6: Performance profiles of the four approaches considered in the computational experiments with instances grouped according to the defined sets.



Chapter 7

Conclusions

The Vehicle Allocation Problem (VAP) bears great applicability in logistics systems, specifically on freight road transportation operations. Given that realistic large-scale instances of the deterministic formulation for this problem have been solved using only heuristic methods (Vasco and Morabito, 2016b), the aim of this work is to propose exact solution methods and models to effectively solve large-scale instances of the VAP.

The first method proposed is Branch-and-Benders-Cut for solving the integer VAP over the space-time network, which consists of adding Benders cuts whenever an integer solution is found along the branch-and-cut tree of a general purpose solver via lazy constraints. The resulting subproblem for each vehicle type from the Benders decomposition is a minimum cost flow problem which bears the integrality property and allows us to use the classical Benders cuts from dual theory. We use the network simplex algorithm for solving the primal subproblem and obtain optimal dual values, and propose a procedure based on network flow algorithms to build the unbounded dual rays when the primal subproblem is infeasible. In addition, we propose two valid inequalities, which are added to the root MP, with the aim of reducing the number of feasibility cuts. We test the proposed algorithm on randomly generated instances and compare its runtime efficiency to the automatic Benders implementation of a general purpose solver as well as the standalone branch-and-cut of the same solver in solving the compact formulation . The results showed that the branch-and-benders-cut algorithm is not yet competitive as all small scale instances were solved faster by the other two methods. Furthermore, we observe convergence problems in the proposed method as none of the instances with 20 to 30 terminals could be solved due to lack of memory considering the other two methods solved all these instances.

The second method proposed is a Branch-and-Price for solving the Dantzig-Wolfe reformulation of the VAP. To this end, we use the work developed in Cruz (2017) which relies on effective procedures for solving each problem of the resulting decomposition and propose four branching schemes to obtain the optimal solution of the integer VAP. Computational experiments consisted of comparing runtime

efficiency and solving capabilities of the proposed algorithm to the branch-and-cut of a general purpose solver in solving the compact formulation of small and realistic large-scale randomly generated instances as well as benchmark instances from the work of Vasco and Morabito (2016b). The results showed the superiority of the proposed algorithm as it was able to solve all large-scale instances faster than the general purpose solver. In addition, it was able to solve almost all instances (except two) while the branch-and-cut solved only 50% of the instances. The main results of this chapter were published in Cruz et al. (2020).

The third approach is a new model that relies on a novel network representation for the problem in which nodes correspond to the requests for transportation services. Different from the ILP models based on the space-time network, the size of the ILP model of this formulation depends exclusively on the number of requests and number of vehicles. Additionally, we propose a branch-and-price algorithm based on the Dantzig-Wolfe reformulation of this new model which relies on efficient procedures such as a stabilized column generation procedure, an efficient algorithm to solve the subproblems and a hierarchical branching scheme. Computational experiments consisted of comparing runtime efficiency and solving capabilities of the general purpose solver in solving the compact formulations of the VAP over the space-time network and the proposed formulation. Also, we compare the runtime efficiency and solving capabilities of the branch-and-price algorithms in solving the Dantzig-Wolfe formulation of the VAP over the space-time network and the Dantzig-Wolfe reformulation based on the proposed formulation. We used the small and realistic large-scale randomly generated instances from Chapter 5 and the benchmark instances from Vasco and Morabito (2016b). The results showed that the proposed formulation performed better in runtime efficiency and solving capabilities when using a general purpose solver than the existing model based on the space-time network. In addition, the proposed branch-and-price proved to be superior in solving the realistic large-scale instances of Vasco and Morabito (2016b) and Chapter 5 than the branch-and-price for solving the reformulation of the VAP based on the space-time network.

In summary, the main contributions of the present work comprises algorithmic and modelling tools for decision makers working in the freight transportation sector facing the problem of how to efficiently reposition empty trucks to serve demand between terminals over a given planning horizon. In previous works, such as Vasco and Morabito (2016b), heuristic methods were used to obtain feasible solutions of realistic large-scale instances representative of continental countries like Brazil. The performance of the solutions obtained in many of those instance were not possible to assess as heuristic methods lacked quality certificates and the size of the resulting models made it impossible for them to be solved with a general purpose solver. In the present work, we showed that by reformulating the problem in such a way that the resulting model depends on different parameters, namely the node-demand formulation,

it is possible to make use of a general purpose solver to solve realistic instances that were not solved using the classical formulation. Furthermore, we also showed that by properly designing exact methods based on decompositions for the classical and proposed formulations, namely the Branch-and-Price, it is possible to optimally solve, or at least obtain quality certificates, for instances where general purpose solvers lack the computational capability to do the same task. Based on the previous considerations, the next section outlines future steps for research on the VAP.

7.1 Future Research

This section outlines two promising directions for future research. The first consist in developing an exact method based on column generation for solving a variant of the VAP that includes fleet sizing decisions. The second consists in developing a compact formulation for this variant based on the node-demand graph representation as well as an exact solution method based on column generation. Following, we present both compact formulations for the VAP with fleet sizing decisions in addition to their respective Dantzig-Wolfe reformulation.

7.1.1 Fleet sizing VAP

Some carriers may not want to reject demand given the future financial loss and deterioration of the customer service level it creates, hence the need to outsource, or postpone if possible, these services to fulfill this demand. To this end, Vasco and Morabito (2016a) proposed an extension of the VAP in which outsourcing fleet is allowed. There are two possibilities for outsourcing services. The first is the one described in the works of Li and Lu (2014); Liu et al. (2010b), in which these services are outsourced to other third-party logistics operator and they have control over the repositioning and allocation process. The other operator sets a price for fulfilling these services and the only decision to be taken is to decide whether to contract these services or not. The second way of outsourcing is to hire independent truck owners, see Vasco (2012); Vasco and Morabito (2016a), who in many cases sets a rent price for using the services during a specified planning horizon. In this case, the decision maker has to decide whether or not to hire vehicles, and if so, decide on how to reposition and allocate these incoming vehicles to attend all demand. Given that the second case poses a greater complexity in managing these systems, we decided to propose extensions of the current work in that line. The model of the fleet sizing VAP as outlined in the second case, writes as

$$\max \quad \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} (p_{ijv} x_{ijtv} - c_{ijv} y_{ijtv})$$

$$- \sum_{i \in N} \sum_{t \in T} \sum_{v \in V} C_{itv} z_{itv} \tag{7.1}$$

s.t. $$\sum_{j \in N} (x_{ijtv} + y_{ijtv}) - \sum_{\substack{j \in N, \\ j \neq i, \\ t > \tau_{ji}}} \left(x_{ji(t-\tau_{ji})v} + y_{ji(t-\tau_{ki})v} \right) \tag{7.2}$$

$$- y_{ii(t-1)v} - z_{itv} = m_{itv}, \qquad \forall i \in N, \forall t \in T, \forall v \in V,$$

$$\sum_{v \in V} x_{ijtv} = d_{ijt}, \qquad \forall i, j \in N, \forall t \in T, \tag{7.3}$$

$$x_{ijtv} = 0 \wedge y_{ijtv} = 0, \; if \; A_{ijv} = 0, \qquad \forall i, j \in N, \forall t \in T, \forall v \in V, \tag{7.4}$$

$$x_{ijtv} \in \mathbb{Z}_+, y_{ijtv} \in \mathbb{Z}_+, \qquad \forall i, j \in N, \forall t \in T, \forall v \in V \tag{7.5}$$

$$z_{itv} \in \mathbb{Z}_+, \qquad \forall i \in N, \forall t \in T, \forall v \in V. \tag{7.6}$$

where z_{itv} and m_{itv} represent the variables and parameters, respectively, quantifying the number of hired and owned vehicles at $i \in N, t \in T$ from vehicle type $v \in V$, and C_{itv} represent the cost of hiring per vehicle type. If we apply the Dantzig-Wolfe decomposition to the linear relaxation of this problem by leaving the demand constraints in the Master Problem (MP), we end up with the following MP

$$\begin{aligned}
\max \quad & \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} p_{ijv} \left(\sum_{q \in Q_v} \lambda_{vq} (\bar{x}_{vqijt}) + \sum_{r \in R_v} \lambda_{vr} (\bar{x}_{vrijt}) \right) \\
& - \sum_{i \in N} \sum_{t \in T} \sum_{v \in V} C_{itv} \left(\sum_{q \in Q_v} \lambda_{vq} (\bar{z}_{vqit}) + \sum_{r \in R_v} \lambda_{vr} (\bar{z}_{vrit}) \right) \\
& - \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} \sum_{t \in T} \sum_{v \in V} c_{ijv} \left(\sum_{q \in Q_v} \lambda_{vq} (\bar{y}_{vqijt}) + \sum_{r \in R_v} \lambda_{vr} (\bar{y}_{vrijt}) \right)
\end{aligned}$$

$$\begin{aligned}
\text{s.t.:} \quad & \sum_{v \in V} \left(\sum_{q \in Q_v} \lambda_{vq} (\bar{x}_{vqijt}) + \sum_{r \in R_v} \lambda_{vr} (\bar{x}_{vrijt}) \right) = d_{ijt}, && \forall i, j \in N, \forall t \in T \quad (u_{ijt}) \\
& \sum_{q \in Q_v} \lambda_{vq} = 1, && \forall v \in V \quad (w_v) \\
& \lambda_{vq} \geq 0, && \forall v \in V, \forall q \in Q_v \\
& \lambda_{vq} \geq 0, && \forall v \in V, \forall r \in R_v
\end{aligned}$$

where Q^v and R^v are the set of extreme points and extreme rays of the sets formed by each vehicle v; u and w represent the vector of dual variables of the coupling and convexity constraints, respectively. Since the MP may have an exponential number of extreme points and rays, columns are iteratively priced out with the following subproblem

$$
\begin{aligned}
Z_{sp}^{v} = \quad & \max \quad \sum_{i\in N}\sum_{\substack{j\in N\\ i\neq j}}\sum_{t\in T}((p_{ijv}-u_{ijt})x_{ijtv}-c_{ijv}y_{ijtv}) \\
& \qquad -\sum_{i\in N}\sum_{t\in T}C_{itv}z_{itv} \\
& \text{s.t.} \quad \sum_{j\in N}(x_{ijtv}+y_{ijtv})-\sum_{\substack{j\in N,\\ j\neq i,\\ t>\tau_{ji}}}\left(x_{ji(t-\tau_{ji})v}+y_{ji(t-\tau_{ki})v}\right) \\
& \qquad -y_{ii(t-1)v}-z_{itv}=m_{itv}, && \forall i\in N, \forall t\in T, \\
& \qquad x_{ijtv}=0\wedge y_{ijtv}=0,\ if\ A_{ijv}=0, && \forall i,j\in N, \forall t\in T, \\
& \qquad x_{ijtv}\in\mathbb{Z}_{+}, y_{ijtv}\in\mathbb{Z}_{+}, && \forall i,j\in N, \forall t\in T, \\
& \qquad z_{itv}\in\mathbb{Z}_{+}, && \forall i\in N, \forall t\in T.
\end{aligned}
$$

where, at a given iteration, a column is likely to be part of the optimal solution if $Z_{sp}^{v}-w_v>0$.

7.1.2 Fleet sizing node-demand VAP

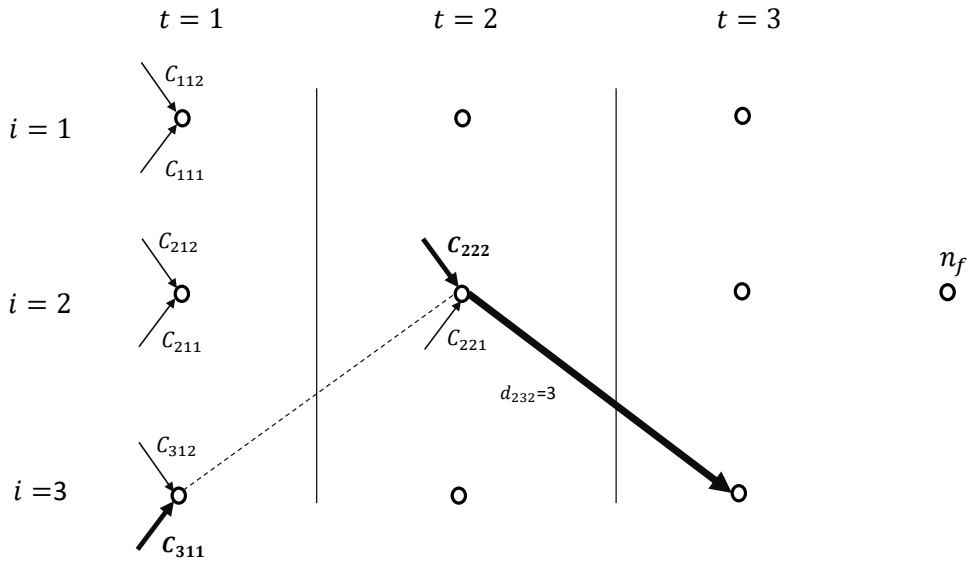


Figure 7.1: Fleet sizing costs.

Another approach for dealing with fleet sizing in empty vehicle repositioning is to extend the node-demand model of Chapter 6 as follows. Given that paths are between requests and artificial depots, we now have to redefine the costs of fleet sizing at a given terminal to attend a given demand. Figure 7.1 shows a situation in which we have one demand arc (d_{232}) and we have to choose hiring between two vehicle types to attend this demand. All possible terminal-period pairs for hiring vehicles are indicated

by small incoming arrows. If there is no limit on the quantity we can hire at a given point for any vehicle type v, it is safe to say that, at an optimal solution, the demand arc is going to be serviced by hired vehicles coming from the same origin whose combined costs of hiring and traveling are the lowest among other terminal-period pairs. For instance, the best option for hiring type-1 vehicles is at $(3, 1)$ and then route them until $(2, 2)$; and the best option for hiring type-2 vehicles is in the same origin of the demand arc. Given this, we can define the cost of hiring vehicles of type v to attend request r as

$$C_{rv} = \min_{(i,t)} \{C_{itv} + \text{ShortestPath}((i, t), (i_r, t_r))\} \tag{7.7}$$

and the whole model writes as

$$\max \quad \sum_{v \in V} \sum_{r \in R \cup \{0\}} \sum_{s \in S \cup \{|R|+1\}} P_{rsv} x_{rsv} - \sum_{v \in V} \sum_{r \in R \cup \{0\}} C_{rv} z_{rv} \tag{7.8}$$

$$\text{s.t.} \quad \sum_{\substack{s \in R \cup |R|+1 \\ s > r}} x_{rsv} = \sum_{\substack{s \in R \cup 0 \\ s < r}} x_{srv} + z_{rv}, \quad \forall r \in R, \forall v \in V \tag{7.9}$$

$$\sum_{s \in R \cup \{|R|+1\}} x_{0sv} = m_v, \quad \forall v \in V \tag{7.10}$$

$$\sum_{r \in R \cup 0} x_{r(|R|+1)v} \geq m_v, \quad \forall v \in V \tag{7.11}$$

$$\sum_{v \in V} \sum_{\substack{s \in R \\ s > r}} x_{rsv} = D_r, \quad \forall r \in R, \tag{7.12}$$

$$z_{rv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R, \forall v \in V \tag{7.13}$$

$$x_{srv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R \cup \{|R| + 1\}, \forall v \in V \tag{7.14}$$

$$x_{rsv} = 0 \quad \text{if} \quad A_{i_r j_r v} = 0, \quad \forall r \in R \cup \{0\}, \forall v \in V \tag{7.15}$$

$$x_{rsv} \in \mathbb{Z}_+, \quad \forall r \in R \cup \{0\}, s \in R \cup \{|R| + 1\}, \forall v \in V \tag{7.16}$$

$$z_{rv} \in \mathbb{Z}_+, \quad \forall r \in R, \forall v \in V \tag{7.17}$$

where z_{rv} denotes the number of vehicles hired from type v to attend request r. Constraints (7.9) enforce flow conservation in the request nodes. Constraints (7.10) and 7.11 ensure m_v vehicles of type v depart from and arrive at depot $\{0\}$ and $|R| + 1$, respectively. Constraints (7.12) ensure all demand at nodes r is covered. Constraints (7.13)-(7.15) restrict ingoing and outgoing flow from nodes that can not be visited by certain vehicle types. Constraints (7.16) and constraints (7.17) are the variables domain.

By applying the Dantzig Wolfe decomposition to the linear relaxation of model (7.8)-(7.15) and leaving constraints (7.12) in the Master Problem, we obtain the following MP

$$\max \quad \sum_{v\in V}\sum_{r\in R\cup\{0\}}\sum_{s\in S\cup\{|R|+1\}} P_{rsv}\left(\sum_{q\in Q_v}\lambda_{qv}\bar{x}_{qv}+\sum_{l\in R_v}\lambda_{lv}\bar{x}_{lv}\right) \tag{7.18}$$

$$\text{s.t.} \quad \sum_{v\in V}\sum_{\substack{s\in R\\ s>r}}\left(\sum_{q\in Q_v}\lambda_{qv}\bar{x}_{qv}+\sum_{l\in R_v}\lambda_{lv}\bar{x}_{lv}\right)=D_r, \qquad \forall r\in R \qquad (u_r) \tag{7.19}$$

$$\sum_{q\in Q_v}\lambda_{qv}=1, \qquad \forall v\in V \qquad (w_v) \tag{7.20}$$

$$\lambda_{qv}\geq 0, \qquad \forall v\in V, \forall q\in Q_v \tag{7.21}$$

where Q^v and R^v are the set of extreme points and extreme rays of the sets formed by each vehicle v; u and w represent the vector of dual variables of the coupling and convexity constraints, respectively. Colums are priced out by solving the following subproblem

$$\begin{aligned}
Z_{sp(v)} = \quad & \max \quad \sum_{r\in R\cup\{0\}}\sum_{s\in S\cup\{|R|+1\}}(P_{rsv}-u_r)x_{rsv}-\sum_{r\in R\cup\{0\}}C_{rv}z_{rv} \\
& \text{s.t.} \quad \sum_{\substack{s\in R\cup|R|+1\\ s>r}}x_{rsv}=\sum_{\substack{s\in R\cup 0\\ s<r}}x_{srv}+z_{rv}, \quad \forall r\in R, \forall v\in V \\
& \qquad \sum_{s\in R\cup\{|R|+1\}}x_{0sv}=m_v, \quad \forall v\in V \\
& \qquad \sum_{r\in R\cup 0}x_{r(|R|+1)v}\geq m_v, \quad \forall v\in V \\
& \qquad \sum_{v\in V}\sum_{\substack{s\in R\\ s>r}}x_{rsv}=D_r, \quad \forall r\in R, \\
& \qquad z_{rv}=0 \quad \text{if} \quad A_{i_rj_rv}=0, \quad \forall r\in R, \forall v\in V \\
& \qquad x_{srv}=0 \quad \text{if} \quad A_{i_rj_rv}=0, \quad \forall r\in R\cup\{|R|+1\}, \forall v\in V \\
& \qquad x_{rsv}=0 \quad \text{if} \quad A_{i_rj_rv}=0, \quad \forall r\in R\cup\{0\}, \forall v\in V \\
& \qquad x_{rsv}\in\mathbb{Z}_+, \quad \forall r\in R\cup\{0\}, s\in R\cup\{|R|+1\}, \forall v\in V \\
& \qquad z_{rv}\in\mathbb{Z}_+, \quad \forall r\in R, \forall v\in V
\end{aligned}$$

and evaluating if their reduced cost $Z^v_{sp}-w_v$ is minor than 0.

In this chapter, we have presented the main scientific contributions of the present work which encompasses a new formulation and exact algorithms based on decomposition techniques in order to solve deterministic formulations of the VAP. In addition, we have introduced a variant of the VAP which models situations in which all demand needs to be serviced within the planning horizon by means of

hiring an independent fleet. Future research based on this work contemplates devising exact algorithms for this new variant and more realistic variants of the VAP.

Appendices

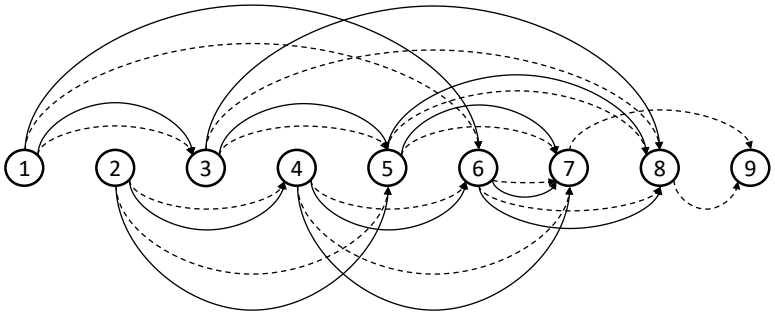
Appendix A

Detailed graphics of the longest path algorithm for the illustrative example of Section 5.3.2

In this section we show graphically every iteration of the longest path algorithm for the illustrative example of Section 5.3.2. Table A.1 contains the travelling costs for empty and loaded arcs for a given type of vehicle. The initial (distance or cost) label of all nodes are $-\infty$ while the label of the starting node 1 is 0. The initial precedence label of all nodes is empty. The bold circles represent the incumbent node at a given iteration (Line 4 of Algorithm 3, while the shaded circles represent the forward adjacent nodes whose labels are being updated in the current bold circle's iteration (Line 8 of Algorithm 3. The bold (either solid or dash) arrows indicate that labels of the adjacent nodes were updated through the distance of the current node. Hence, if at a given iteration there is no bold (solid or dash) arrow, it means the label of the adjacent nodes are not updated (since the current label is the maximum distance according to the optimality conditions in Line 7 of Algorithm 3).

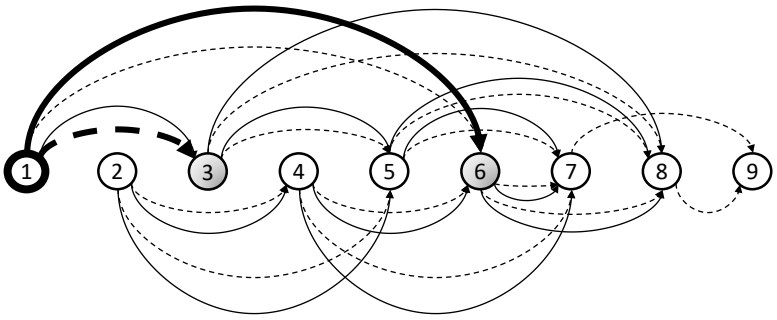
Arc	1-3	1-6	2-4	2-5	3-5	3-8	4-6	4-7	5-7	5-8	6-7	6-8	7-9	8-9
Empty cost	0,00	-2,50	0,00	-1,50	0,00	-1,00	0,00	-2,20	-2,00	-1,00	0,00	0,00	0,00	0,00
Loaded cost	-inf	2,00	-inf	-2,50	-inf	-2,00	-inf	1,50	2,00	2,50	-inf	-inf	-inf	-inf

Table A.1: Arcs' costs for the ilustrative example



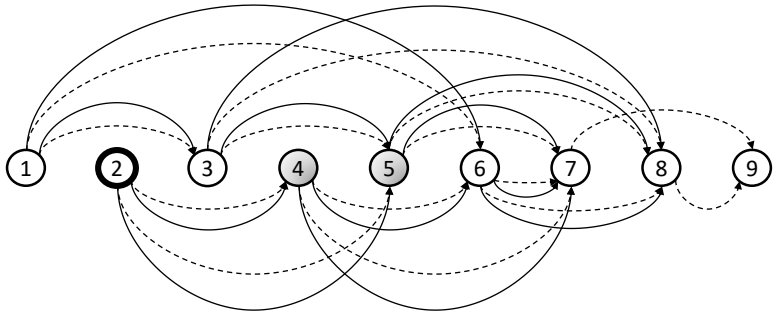
d[1]=0	d[2]=-inf	d[3]=-inf	d[4]=-inf	d[5]=-inf	d[6]=-inf	d[7]=-inf	d[8]=-inf	d[9]=-inf
p[1]=null	p[2]=null	p[3]=null	p[4]=null	p[5]=null	p[6]=null	p[7]=null	p[8]=null	p[9]=null

Figure A.1: Initialization of Maximum Path



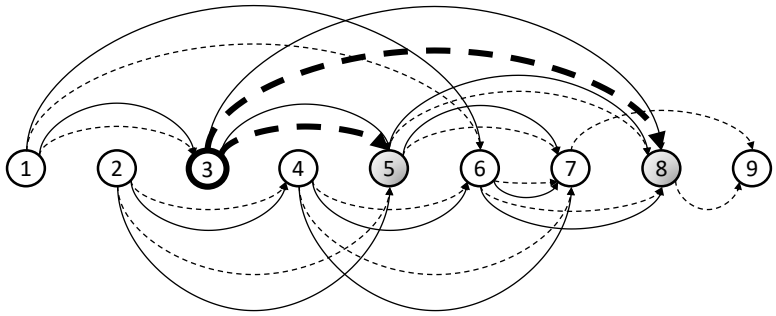
d[1]=0	d[2]=-inf	**d[3]=0**	d[4]=-inf	d[5]=-inf	**d[6]=2**	d[7]=-inf	d[8]=-inf	d[9]=-inf
p[1]=null	p[2]=null	**p[3]=1**	p[4]=null	p[5]=null	**p[6]=1**	p[7]=null	p[8]=null	p[9]=null

Figure A.2: Updating labels adjacent to node 1



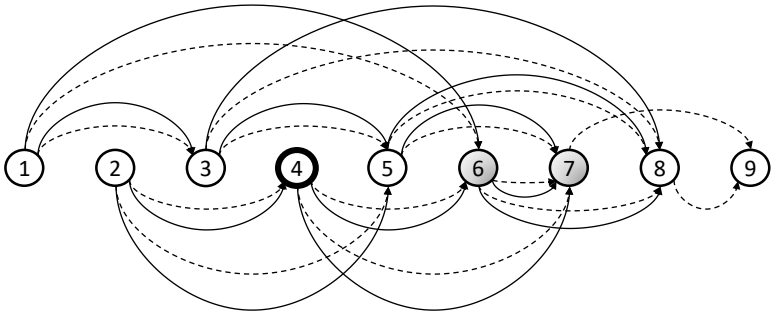
d[1]=0	**d[2]=-inf**	d[3]=0	**d[4]=-inf**	**d[5]=-inf**	d[6]=2	d[7]=-inf	d[8]=-inf	d[9]=-inf
p[1]=null	**p[2]=null**	p[3]=1	**p[4]=null**	**p[5]=null**	p[6]=1	p[7]=null	p[8]=null	p[9]=null

Figure A.3: Updating labels adjacent to node 2



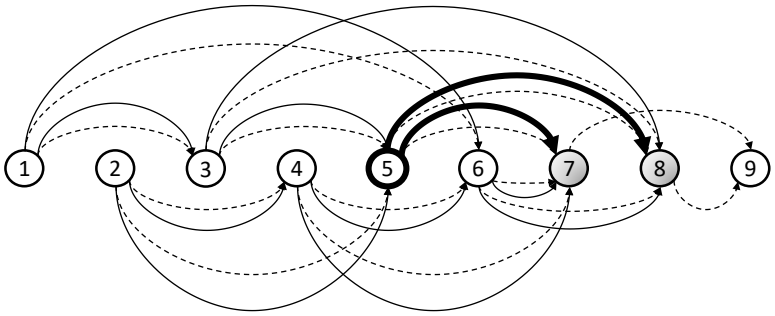
d[1]=0	d[2]=-inf	**d[3]=0**	d[4]=-inf	**d[5]=0**	d[6]=2	d[7]=-inf	**d[8]=-1**	d[9]=-inf
p[1]=null	p[2]=null	**p[3]=1**	p[4]=null	**p[5]=3**	p[6]=1	p[7]=null	**p[8]=3**	p[9]=null

Figure A.4: Updating labels adjacent to node 3



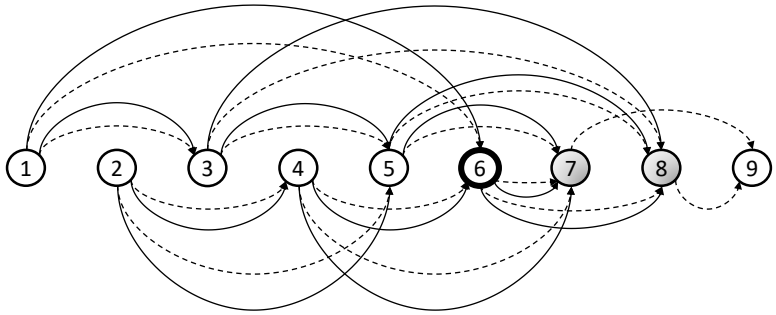
d[1]=0	d[2]=-inf	d[3]=0	**d[4]=-inf**	d[5]=0	**d[6]=2**	**d[7]=-inf**	d[8]=-1	d[9]=-inf
p[1]=null	p[2]=null	p[3]=1	**p[4]=null**	p[5]=3	**p[6]=1**	**p[7]=null**	p[8]=3	p[9]=null

Figure A.5: Updating labels adjacent to node 4



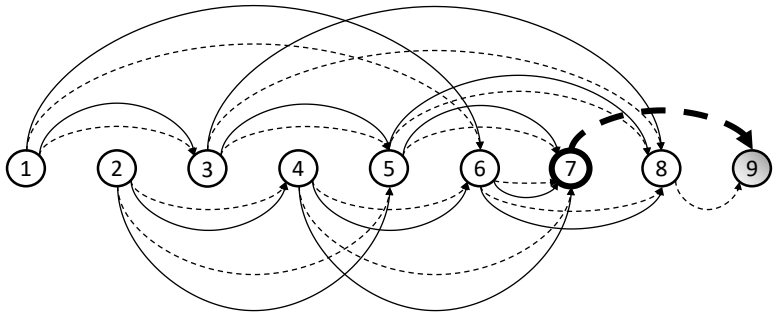
d[1]=0	d[2]=-inf	d[3]=0	d[4]=-inf	**d[5]=0**	d[6]=2	**d[7]=2**	**d[8]=2.5**	d[9]=-inf
p[1]=null	p[2]=null	p[3]=1	p[4]=null	**p[5]=3**	p[6]=1	**p[7]=5**	**p[8]=5**	p[9]=null

Figure A.6: Updating labels adjacent to node 5



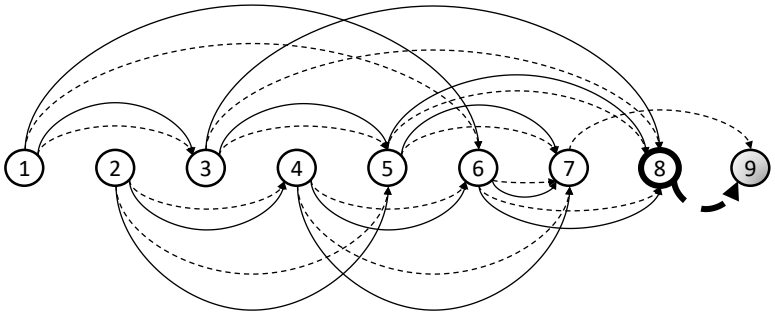
d[1]=0	d[2]=-inf	d[3]=0	d[4]=-inf	d[5]=0	**d[6]=2**	**d[7]=2**	**d[8]=2.5**	d[9]=-inf
p[1]=null	p[2]=null	p[3]=1	p[4]=null	p[5]=3	**p[6]=1**	**p[7]=5**	**p[8]=5**	p[9]=null

Figure A.7: Updating labels adjacent to node 6



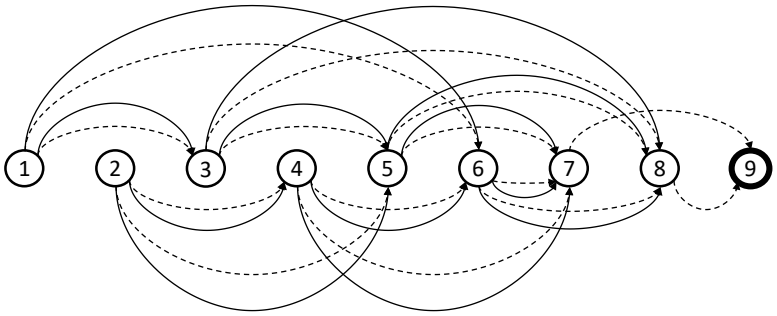
d[1]=0	d[2]=-inf	d[3]=0	d[4]=-inf	d[5]=0	d[6]=2	**d[7]=2**	d[8]=2.5	**d[9]=2**
p[1]=null	p[2]=null	p[3]=1	p[4]=null	p[5]=3	p[6]=1	**p[7]=5**	p[8]=5	**p[9]=7**

Figure A.8: Updating labels adjacent to node 7



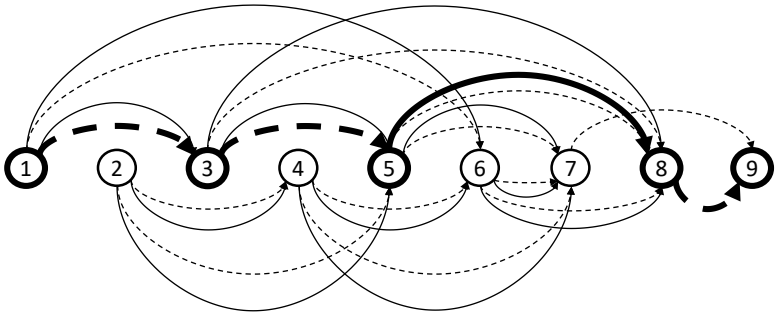
d[1]=0	d[2]=-inf	d[3]=0	d[4]=-inf	d[5]=0	d[6]=2	d[7]=2	**d[8]=2.5**	**d[9]=2.5**
p[1]=null	p[2]=null	p[3]=1	p[4]=null	p[5]=3	p[6]=1	p[7]=5	**p[8]=5**	**p[9]=8**

Figure A.9: Updating labels adjacent to node 8



d[1]=0	d[2]=-inf	d[3]=0	d[4]=-inf	d[5]=0	d[6]=2	d[7]=2	d[8]=2.5	**d[9]=2.5**
p[1]=null	p[2]=null	p[3]=1	p[4]=null	p[5]=3	p[6]=1	p[7]=5	p[8]=5	**p[9]=8**

Figure A.10: Updating labels adjacent to node 9 (no forward adjacent nodes)



d[1]=0	d[2]=-inf	**d[3]=0**	d[4]=-inf	**d[5]=0**	d[6]=2	d[7]=2	**d[8]=2.5**	**d[9]=2.5**
p[1]=null	p[2]=null	**p[3]=1**	p[4]=null	**p[5]=3**	p[6]=1	p[7]=5	**p[8]=5**	**p[9]=8**

Figure A.11: Longest path between node 1 and 9

Appendix B

Results of the Branch-and-Price for the network flow formulation of the VAP

This appendix shows the results of solving 200 randomly generated instances with integrality gap in order to test the proposed Branch-and-Price method. A sign "*" indicates that CPLEX could not solve or even mount the model due to lack of computer memory. A time limit of 3600 seconds was imposed on each run. The following tables contain information that allow us to draw conclusions on the performance of our proposed method compared to CPLEX. Columns refer to:

- Instance is the name of the instance (number of terminals - number of periods - number of vehicles - number of requests.)
- LP is the optimal value of the LP relaxation of model (2.1)–(2.5).
- CPU LP (sec) is the time taken by CPLEX to solve each LP instance to optimality.
- IP is the optimal ILP value of model (2.1)–(2.5).
- CPU IP (sec) is the time taken by CPLEX to solve each ILP instance to optimality.
- CPU PD (sec) is the time taken by the PDCGM to solve the MP (5.1)–(5.4).
- CPU BP-α is the time taken by the B&P to solve each ILP instance to optimality with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- N.T-α is total number of nodes created with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- N.E-α is total number of nodes explored with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- UB BP-α is the upperbound reached with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.

- LB BP-α is the lowerbound reached with the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.
- GAP-α is the relative gap between UB BP-α and LB BP-α when using the branching scheme $\alpha \in \{\text{A,B,C,D}\}$.

Table B.1: Results of the small-scale instances (number of terminals in the range [10,14]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
10-10-20-20	1940	0.13	1937	0.52	0.09	0.16	3	3	1937	1937	0.00%	0.16	3	3	1937	1937	0.00%
10-10-20-25	1874	0.06	1857	0.12	0.06	0.24	5	5	1857	1857	0.00%	0.69	15	15	1857	1857	0.00%
10-10-20-30	507	0.05	501	0.10	0.06	0.92	19	19	501	501	0.00%	0.14	3	3	501	501	0.00%
10-10-20-35	2474.5	0.05	2464	0.16	0.07	0.40	7	7	2464	2464	0.00%	0.36	7	7	2464	2464	0.00%
10-10-20-50	422.5	0.07	421	0.15	0.07	1.03	9	9	421	421	0.00%	0.23	3	3	421	421	0.00%
11-10-20-20	1601	0.06	1584	0.50	0.09	0.20	3	3	1584	1584	0.00%	0.45	7	7	1584	1584	0.00%
11-10-20-25	661.5	0.07	659	0.13	0.09	0.49	7	7	659	659	0.00%	0.19	3	3	659	659	0.00%
11-10-20-30	1606.5	0.06	1596	0.16	0.06	0.18	3	3	1596	1596	0.00%	0.17	3	3	1596	1596	0.00%
11-10-20-35	2240.5	0.06	2238	0.29	0.07	0.32	5	5	2238	2238	0.00%	0.18	3	3	2238	2238	0.00%
11-10-20-50	2704.5	0.08	2701	0.26	0.37	1.94	15	15	2701	2701	0.00%	0.73	5	5	2701	2701	0.00%
12-10-20-20	907	0.08	904	0.59	0.16	11.72	163	163	904	904	0.00%	0.36	3	3	904	904	0.00%
12-10-20-25	396.5	0.08	396	0.21	0.09	0.69	7	7	396	396	0.00%	0.42	5	5	396	396	0.00%
12-10-20-30	337	0.08	335	0.17	0.11	3.06	31	31	335	335	0.00%	0.65	7	7	335	335	0.00%
12-10-20-35	677.5	0.08	669	0.18	0.13	12.27	127	127	669	669	0.00%	2.57	27	27	669	669	0.00%
12-10-20-50	811.5	0.08	811	0.16	0.10	0.26	3	3	811	811	0.00%	0.21	3	3	811	811	0.00%
13-10-20-20	1259.5	0.10	1259	0.18	0.12	1.43	13	13	1259	1259	0.00%	0.81	7	7	1259	1259	0.00%
13-10-20-25	671.5	0.09	669	0.26	0.12	0.67	7	7	669	669	0.00%	0.48	5	5	669	669	0.00%
13-10-20-30	581	0.09	580	0.17	0.18	7.15	53	53	580	580	0.00%	0.32	3	3	580	580	0.00%
13-10-20-35	267.5	0.09	266	0.16	0.10	0.32	3	3	266	266	0.00%	0.28	3	3	266	266	0.00%
13-10-20-50	680	0.09	678	0.33	0.11	0.62	7	7	678	678	0.00%	0.55	7	7	678	678	0.00%
14-10-20-20	636.5	0.11	633	0.21	0.16	0.59	5	5	633	633	0.00%	1.16	7	7	633	633	0.00%
14-10-20-25	1295	0.12	1292	0.36	0.11	10.47	71	71	1292	1292	0.00%	2.00	19	19	1292	1292	0.00%
14-10-20-30	579	0.11	574	0.31	0.13	0.75	7	7	574	574	0.00%	0.33	3	3	574	574	0.00%
14-10-20-35	2058.5	0.11	2057	0.21	0.13	0.86	9	9	2057	2057	0.00%	0.29	3	3	2057	2057	0.00%
14-10-20-50	356	0.11	355	0.65	0.13	2.08	17	17	355	355	0.00%	0.29	3	3	355	355	0.00%

Table B.2: Results of the small-scale instances (number of terminals in the range [10,14]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
10-10-20-20	1940	0.13	1937	0.52	0.09	0.18	3	3	1937	1937	0.00%	0.18	3	3	1937	1937	0.00%
10-10-20-25	1874	0.06	1857	0.12	0.06	0.15	3	3	1857	1857	0.00%	0.15	3	3	1857	1857	0.00%
10-10-20-30	507	0.05	501	0.10	0.06	0.59	9	9	501	501	0.00%	0.44	9	9	501	501	0.00%
10-10-20-35	2474.5	0.05	2464	0.16	0.07	0.19	3	3	2464	2464	0.00%	0.18	3	3	2464	2464	0.00%
10-10-20-50	422.5	0.07	421	0.15	0.07	0.31	3	3	421	421	0.00%	0.15	3	3	421	421	0.00%
11-10-20-20	1601	0.06	1584	0.50	0.09	0.18	3	3	1584	1584	0.00%	0.19	3	3	1584	1584	0.00%
11-10-20-25	661.5	0.07	659	0.13	0.09	0.22	3	3	659	659	0.00%	0.18	3	3	659	659	0.00%
11-10-20-30	1606.5	0.06	1596	0.16	0.06	0.18	3	3	1596	1596	0.00%	0.18	3	3	1596	1596	0.00%
11-10-20-35	2240.5	0.06	2238	0.29	0.07	0.22	3	3	2238	2238	0.00%	0.20	3	3	2238	2238	0.00%
11-10-20-50	2704.5	0.08	2701	0.26	0.37	0.46	3	3	2701	2701	0.00%	0.23	3	3	2701	2701	0.00%
12-10-20-20	907	0.08	904	0.59	0.16	1.32	9	9	904	904	0.00%	0.76	9	9	904	904	0.00%
12-10-20-25	396.5	0.08	396	0.21	0.09	0.24	3	3	396	396	0.00%	0.25	3	3	396	396	0.00%
12-10-20-30	337	0.08	335	0.17	0.11	1.04	7	7	335	335	0.00%	0.65	7	7	335	335	0.00%
12-10-20-35	677.5	0.08	669	0.18	0.13	3.30	23	23	669	669	0.00%	1.79	23	23	669	669	0.00%
12-10-20-50	811.5	0.08	811	0.16	0.10	0.26	3	3	811	811	0.00%	0.25	3	3	811	811	0.00%
13-10-20-20	1259.5	0.10	1259	0.18	0.12	0.27	3	3	1259	1259	0.00%	0.27	3	3	1259	1259	0.00%
13-10-20-25	671.5	0.09	669	0.26	0.12	0.29	3	3	669	669	0.00%	0.27	3	3	669	669	0.00%
13-10-20-30	581	0.09	580	0.17	0.18	0.35	3	3	580	580	0.00%	0.29	3	3	580	580	0.00%
13-10-20-35	267.5	0.09	266	0.16	0.10	0.89	9	9	266	266	0.00%	0.80	9	9	266	266	0.00%
13-10-20-50	680	0.09	678	0.33	0.11	0.45	5	5	678	678	0.00%	0.42	5	5	678	678	0.00%
14-10-20-20	636.5	0.11	633	0.21	0.16	1.21	5	5	633	633	0.00%	0.56	5	5	633	633	0.00%
14-10-20-25	1295	0.12	1292	0.36	0.11	0.39	3	3	1292	1292	0.00%	0.35	3	3	1292	1292	0.00%
14-10-20-30	579	0.11	574	0.31	0.13	0.31	3	3	574	574	0.00%	0.30	3	3	574	574	0.00%
14-10-20-35	2058.5	0.11	2057	0.21	0.13	0.29	3	3	2057	2057	0.00%	0.29	3	3	2057	2057	0.00%
14-10-20-50	356	0.11	355	0.65	0.13	0.31	3	3	355	355	0.00%	0.31	3	3	355	355	0.00%

Table B.3: Results of the small-scale instances (number of terminals in the range [15,19]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
15-10-20-20	564	0.13	560	0.26	0.14	2.09	19	19	560	560	0.00%	0.60	3	3	560	560	0.00%
15-10-20-25	1202.5	0.13	1202	0.27	0.32	0.67	5	5	1202	1202	0.00%	0.44	3	3	1202	1202	0.00%
15-10-20-30	1454.5	0.12	1451	0.26	0.11	0.93	5	5	1451	1451	0.00%	0.29	3	3	1451	1451	0.00%
15-10-20-35	1679.5	0.12	1679	0.23	0.15	0.45	3	3	1679	1679	0.00%	0.31	3	3	1679	1679	0.00%
15-10-20-50	624	0.13	623	0.36	0.15	0.38	3	3	623	623	0.00%	0.37	3	3	623	623	0.00%
16-10-20-20	738	0.14	736	0.46	0.32	4.62	27	27	736	736	0.00%	1.25	7	7	736	736	0.00%
16-10-20-25	1481.5	0.15	1479	0.29	0.16	0.86	5	5	1479	1479	0.00%	0.36	3	3	1479	1479	0.00%
16-10-20-30	1800	0.14	1788	0.28	0.16	2.78	19	19	1788	1788	0.00%	0.44	3	3	1788	1788	0.00%
16-10-20-35	2040.5	0.14	2037	0.27	0.14	0.55	5	5	2037	2037	0.00%	0.33	3	3	2037	2037	0.00%
16-10-20-50	1564.5	0.15	1562	0.33	0.16	0.89	7	7	1562	1562	0.00%	0.35	3	3	1562	1562	0.00%
17-10-20-20	632.5	0.16	632	0.30	0.24	1.09	5	5	632	632	0.00%	0.59	3	3	632	632	0.00%
17-10-20-25	1613.5	0.16	1611	0.31	0.18	0.82	7	7	1611	1611	0.00%	0.38	3	3	1611	1611	0.00%
17-10-20-30	424.5	0.16	424	0.28	0.18	0.96	3	3	424	424	0.00%	0.55	3	3	424	424	0.00%
17-10-20-35	2349	0.17	2347	0.32	0.24	1.12	5	5	2347	2347	0.00%	0.61	3	3	2347	2347	0.00%
17-10-20-50	2210	0.16	2202	0.37	0.17	0.38	3	3	2202	2202	0.00%	0.40	3	3	2202	2202	0.00%
18-10-20-20	1147.5	0.17	1143	0.29	0.31	17.18	53	53	1143	1143	0.00%	0.87	3	3	1143	1143	0.00%
18-10-20-25	294.5	0.17	291	0.41	0.23	0.64	3	3	291	291	0.00%	0.56	3	3	291	291	0.00%
18-10-20-30	1836	0.17	1835	0.29	0.25	0.93	5	5	1835	1835	0.00%	0.53	3	3	1835	1835	0.00%
18-10-20-35	992	0.18	991	0.78	0.18	1.39	9	9	991	991	0.00%	0.56	3	3	991	991	0.00%
18-10-20-50	550.5	0.20	550	0.35	0.22	0.48	3	3	550	550	0.00%	0.58	3	3	550	550	0.00%
19-10-20-20	906.5	0.19	905	0.33	0.25	0.54	3	3	905	905	0.00%	0.60	3	3	905	905	0.00%
19-10-20-25	535.5	0.20	535	0.36	0.28	0.77	3	3	535	535	0.00%	0.60	3	3	535	535	0.00%
19-10-20-30	1405.5	0.22	1402	0.45	0.18	0.60	3	3	1402	1402	0.00%	0.46	3	3	1402	1402	0.00%
19-10-20-35	692.5	0.20	691	0.38	0.22	0.57	3	3	691	691	0.00%	0.59	3	3	691	691	0.00%
19-10-20-50	2599	0.21	2591	0.61	0.23	1.21	7	7	2591	2591	0.00%	1.05	5	5	2591	2591	0.00%

Table B.4: Results of the small-scale instances (number of terminals in the range [15,19]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
15-10-20-20	564	0.13	560	0.26	0.14	0.64	3	3	560	560	0.00%	0.37	3	3	560	560	0.00%
15-10-20-25	1202.5	0.13	1202	0.27	0.32	0.68	5	5	1202	1202	0.00%	0.57	5	5	1202	1202	0.00%
15-10-20-30	1454.5	0.12	1451	0.26	0.11	0.33	3	3	1451	1451	0.00%	0.32	3	3	1451	1451	0.00%
15-10-20-35	1679.5	0.12	1679	0.23	0.15	0.33	3	3	1679	1679	0.00%	0.30	3	3	1679	1679	0.00%
15-10-20-50	624	0.13	623	0.36	0.15	0.44	3	3	623	623	0.00%	0.41	3	3	623	623	0.00%
16-10-20-20	738	0.14	736	0.46	0.32	0.68	3	3	736	736	0.00%	0.52	3	3	736	736	0.00%
16-10-20-25	1481.5	0.15	1479	0.29	0.16	0.40	3	3	1479	1479	0.00%	0.38	3	3	1479	1479	0.00%
16-10-20-30	1800	0.14	1788	0.28	0.16	0.44	3	3	1788	1788	0.00%	0.37	3	3	1788	1788	0.00%
16-10-20-35	2040.5	0.14	2037	0.27	0.14	0.44	3	3	2037	2037	0.00%	0.42	3	3	2037	2037	0.00%
16-10-20-50	1564.5	0.15	1562	0.33	0.16	0.47	3	3	1562	1562	0.00%	0.40	3	3	1562	1562	0.00%
17-10-20-20	632.5	0.16	632	0.30	0.24	0.49	3	3	632	632	0.00%	0.47	3	3	632	632	0.00%
17-10-20-25	1613.5	0.16	1611	0.31	0.18	0.98	7	7	1611	1611	0.00%	0.87	7	7	1611	1611	0.00%
17-10-20-30	424.5	0.16	424	0.28	0.18	0.83	3	3	424	424	0.00%	0.48	3	3	424	424	0.00%
17-10-20-35	2349	0.17	2347	0.32	0.24	0.56	3	3	-2346.99	2346.99	0.00%	0.43	3	3	2346.99	2346.99	0.00%
17-10-20-50	2210	0.16	2202	0.37	0.17	0.67	5	5	2202	2202	0.00%	0.63	5	5	2202	2202	0.00%
18-10-20-20	1147.5	0.17	1143	0.29	0.31	1.16	5	5	1143	1143	0.00%	0.91	5	5	1143	1143	0.00%
18-10-20-25	294.5	0.17	291	0.41	0.23	0.65	3	3	291	291	0.00%	0.59	3	3	291	291	0.00%
18-10-20-30	1836	0.17	1835	0.29	0.25	0.96	5	5	1835	1835	0.00%	0.81	5	5	1835	1835	0.00%
18-10-20-35	992	0.18	991	0.78	0.18	0.46	3	3	991	991	0.00%	0.45	3	3	991	991	0.00%
18-10-20-50	550.5	0.20	550	0.35	0.22	0.59	3	3	550	550	0.00%	0.51	3	3	550	550	0.00%
19-10-20-20	906.5	0.19	905	0.33	0.25	0.58	3	3	905	905	0.00%	0.50	3	3	905	905	0.00%
19-10-20-25	535.5	0.20	535	0.36	0.28	0.67	3	3	535	535	0.00%	0.62	3	3	535	535	0.00%
19-10-20-30	1405.5	0.22	1402	0.45	0.18	0.91	7	5	1402	1402	0.00%	0.83	7	5	1402	1402	0.00%
19-10-20-35	692.5	0.20	691	0.38	0.22	0.52	3	3	691	691	0.00%	0.49	3	3	691	691	0.00%
19-10-20-50	2599	0.21	2591	0.61	0.23	1.21	5	5	2591	2591	0.00%	0.86	5	5	2591	2591	0.00%

Table B.5: Results of the small-scale instances (number of terminals in the range [20,29]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
20-20-100-200	14473	3.49	14471	16.23	1.48	3599.07	4815	4783	14471.5	14471	0.00%	671.16	989	989	14471	14471	0.00%
20-20-130-200	31331	4.33	31327	19.13	1.51	3599.58	4361	4091	31327.5	31327	0.00%	3599.83	5517	3905	31329.5	31327	0.01%
20-20-150-200	34622.5	5.45	34618	22.10	1.70	3599.37	4623	3406	34622.5	34618	0.01%	56.30	63	63	34618	34618	0.00%
21-20-100-200	2585.5	5.36	2585	19.09	1.75	3599.85	4877	3738	2585.5	2585	0.02%	14.56	15	15	2585	2585	0.00%
21-20-130-200	4056.5	5.01	4056	21.65	2.05	3599.09	4567	3661	4056.5	4056	0.01%	3599.17	4695	3885	4056.5	4056	0.01%
21-20-150-200	27333.5	5.51	27322	23.34	1.88	7.38	7	7	27322	27322	0.00%	175.71	185	185	27322	27322	0.00%
22-20-100-200	4902.5	5.82	4902	23.76	1.96	86.48	95	95	4902	4902	0.00%	13.49	13	13	4902	4902	0.00%
22-20-130-200	12304	5.71	12301	25.56	1.99	3599.01	5275	3282	12304	12301	0.02%	14.68	15	15	12301	12301	0.00%
22-20-150-200	10972.5	7.14	10971	34.33	2.12	3599.65	4279	2882	10972.5	10971	0.01%	3599.46	3841	2530	10972.5	10971	0.01%
23-20-100-200	14133.5	4.65	14132	19.48	1.90	3599.61	4299	3039	14133.5	14132	0.01%	1126.82	1067	1067	14132	14132	0.00%
23-20-130-200	21749.5	6.44	21745	27.30	1.94	387.39	389	389	21745	21745	0.00%	171.37	159	159	21745	21745	0.00%
23-20-150-200	23366	8.00	23358	31.67	2.20	3599.11	4083	2499	23366	23358	0.03%	3599.11	4465	2605	23366	23358	0.03%
24-20-100-200	10157	5.30	10155	24.04	2.37	4.98	3	3	10155	10155	0.00%	22.78	21	21	10155	10155	0.00%
24-20-130-200	3486.5	11.08	3486	42.33	3.51	3599.57	3193	2020	3486.5	3486	0.01%	3600.44	2503	2375	3486.5	3486	0.01%
24-20-150-200	18158	12.68	18156	52.91	3.26	3600.64	4105	2259	18158	18156	0.01%	3600.07	3541	2631	18158	18156	0.01%
25-20-100-200	9336.5	5.20	9335	23.05	2.33	133.04	123	123	9335	9335	0.00%	9.31	7	7	9335	9335	0.00%
25-20-130-200	33906	8.86	33905	38.92	2.64	3599.42	4343	2783	33906	33905	0.00%	13.09	11	11	33905	33905	0.00%
25-20-150-200	32517	9.41	32508	37.72	2.91	3599.39	3765	2074	32517	32508	0.03%	3599.89	3603	2036	32517	32508	0.03%
26-20-100-200	9816	5.49	9815	22.13	2.64	163.62	145	145	9815	9815	0.00%	17.65	15	15	9815	9815	0.00%
26-20-130-200	16895	7.66	16892	33.17	2.76	3599.49	3439	2492	16895	16892	0.02%	9.97	7	7	16892	16892	0.00%
26-20-150-200	3731.5	19.33	3731	67.78	3.94	3600.89	2679	1894	3731.5	3731	0.01%	3600.75	2927	1825	3731.5	3731	0.01%
27-20-100-200	21095.5	7.05	21090	32.24	3.31	3599.80	3471	2688	21094.5	21090	0.02%	5.24	3	3	21090	21090	0.00%
27-20-130-200	8457.5	11.71	8456	55.14	3.95	3600.25	2945	1670	8457.5	8456	0.02%	3599.23	2367	1973	8457	8456	0.01%
27-20-150-200	26709	9.95	26704	44.40	3.29	3599.21	3155	2244	26709	26704	0.02%	536.04	389	389	26704	26704	0.00%
28-20-100-200	3641.5	10.32	3641	42.01	3.23	6.87	3	3	3641	3641	0.00%	6.86	3	3	3641	3641	0.00%
28-20-130-200	25359	8.64	25355	36.32	3.32	3600.63	3067	2188	25359	25355	0.02%	6.23	3	3	25355	25355	0.00%
28-20-150-200	11225.5	11.36	11222	57.47	3.73	3599.85	2835	1717	11225.5	11222	0.03%	3599.45	2349	2172	11225.5	11222	0.03%
29-20-100-200	5995	7.09	5994	31.58	3.69	3599.60	2915	2215	5995	5994	0.02%	3599.60	2641	2462	5994.5	5994	0.01%
29-20-130-200	8979.5	11.48	8978	49.86	3.60	15.50	9	9	8978	8978	0.00%	3599.29	2605	1994	8979.5	8978	0.02%
29-20-150-200	7978.5	10.10	7977	44.10	3.91	3600.50	2779	1796	7978.5	7977	0.02%	3599.24	2181	2025	7978.5	7977	0.02%

Table B.6: Results of the small-scale instances (number of terminals in the range [20,29]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
20-20-100-200	14473	3.49	14471	16.23	1.48	6.63	7	7	14471	14471	0.00%	5.36	7	7	14471	14471	0.00%
20-20-130-200	31331	4.33	31327	19.13	1.51	9.42	9	9	31327	31327	0.00%	7.51	9	9	31327	31327	0.00%
20-20-150-200	34622.5	5.45	34618	22.10	1.70	3.01	3	3	34618	34618	0.00%	2.63	3	3	34618	34618	0.00%
21-20-100-200	2585.5	5.36	2585	19.09	1.75	3.76	3	3	2585	2585	0.00%	3.09	3	3	2585	2585	0.00%
21-20-130-200	4056.5	5.01	4056	21.65	2.05	4.02	3	3	4056	4056	0.00%	3.51	3	3	4056	4056	0.00%
21-20-150-200	27333.5	5.51	27322	23.34	1.88	4.18	3	3	27322	27322	0.00%	3.56	3	3	27322	27322	0.00%
22-20-100-200	4902.5	5.82	4902	23.76	1.96	6.90	5	5	4902	4902	0.00%	5.71	5	5	4902	4902	0.00%
22-20-130-200	12304	5.71	12301	25.56	1.99	4.72	3	3	12301	12301	0.00%	4.04	3	3	12301	12301	0.00%
22-20-150-200	10972.5	7.14	10971	34.33	2.12	7.16	5	5	10971	10971	0.00%	5.47	5	5	10971	10971	0.00%
23-20-100-200	14133.5	4.65	14132	19.48	1.90	3.75	3	3	14132	14132	0.00%	2.99	3	3	14132	14132	0.00%
23-20-130-200	21749.5	6.44	21745	27.30	1.94	3.92	3	3	21745	21745	0.00%	3.02	3	3	21745	21745	0.00%
23-20-150-200	23366	8.00	23358	31.67	2.20	5.87	5	5	23358	23358	0.00%	5.03	5	5	23358	23358	0.00%
24-20-100-200	10157	5.30	10155	24.04	2.37	8.00	5	5	10155	10155	0.00%	6.28	5	5	10155	10155	0.00%
24-20-130-200	3486.5	11.08	3486	42.33	3.51	17.85	9	9	3486	3486	0.00%	13.02	9	9	3486	3486	0.00%
24-20-150-200	18158	12.68	18156	52.91	3.26	12.45	7	7	18156	18156	0.00%	9.43	7	7	18156	18156	0.00%
25-20-100-200	9336.5	5.20	9335	23.05	2.33	7.10	5	5	9335	9335	0.00%	5.93	5	5	9335	9335	0.00%
25-20-130-200	33906	8.86	33905	38.92	2.64	4.65	3	3	33905	33905	0.00%	4.07	3	3	33905	33905	0.00%
25-20-150-200	32517	9.41	32508	37.72	2.91	6.87	5	5	32508	32508	0.00%	6.05	5	5	32508	32508	0.00%
26-20-100-200	9816	5.49	9815	22.13	2.64	4.37	3	3	9815	9815	0.00%	4.16	3	3	9815	9815	0.00%
26-20-130-200	16895	7.66	16892	33.17	2.76	4.20	3	3	16892	16892	0.00%	3.93	3	3	16892	16892	0.00%
26-20-150-200	3731.5	19.33	3731	67.78	3.94	24.24	11	11	3731	3731	0.00%	21.92	11	11	3731	3731	0.00%
27-20-100-200	21095.5	7.05	21090	32.24	3.31	4.93	3	3	21090	21090	0.00%	4.54	3	3	21090	21090	0.00%
27-20-130-200	8457.5	11.71	8456	55.14	3.95	33.61	19	19	8456	8456	0.00%	29.05	19	19	8456	8456	0.00%
27-20-150-200	26709	9.95	26704	44.40	3.29	7.41	5	5	26704	26704	0.00%	6.69	5	5	26704	26704	0.00%
28-20-100-200	3641.5	10.32	3641	42.01	3.23	47.20	29	29	3641	3641	0.00%	42.24	29	29	3641	3641	0.00%
28-20-130-200	25359	8.64	25355	36.32	3.32	5.14	3	3	25355	25355	0.00%	4.63	3	3	25355	25355	0.00%
28-20-150-200	11225.5	11.36	11222	57.47	3.73	6.16	3	3	11222	11222	0.00%	5.49	3	3	11222	11222	0.00%
29-20-100-200	5995	7.09	5994	31.58	3.69	8.81	5	5	5994	5994	0.00%	7.48	5	5	5994	5994	0.00%
29-20-130-200	8979.5	11.48	8978	49.86	3.60	80.44	49	49	8978	8978	0.00%	72.53	49	49	8978	8978	0.00%
29-20-150-200	7978.5	10.10	7977	44.10	3.91	7.37	3	3	7977	7977	0.00%	6.73	3	3	7977	7977	0.00%

Table B.7: Results of the large-scale instances (number of terminals in the range [30,39]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
30-30-200-300	14513.5	100.44	14512	392.92	8.23	3599.63	1347	754	14513.5	14512	0.01%	3599.05	1105	824	14513	14512	0.01%
30-30-230-300	47060.5	104.13	47054	435.42	9.41	3600.03	1215	662	47060.5	47054	0.01%	2863.61	759	759	47054	47054	0.00%
30-30-250-300	69007	153.42	68998	750.87	10.03	3599.86	1273	660	69007	68998	0.01%	3602.32	1253	629	69007	68998	0.01%
31-30-200-300	51227.5	98.37	51221	436.45	8.46	3600.81	1223	770	51227.5	51221	0.01%	186.57	47	47	51221	51221	0.00%
31-30-230-300	79462.5	135.00	79441	716.02	9.68	3602.18	1225	661	79462.5	79441	0.03%	3600.01	1117	678	79462.5	79441	0.03%
31-30-250-300	79072	139.71	79063	685.50	10.34	3600.23	1113	620	79072	79063	0.01%	3602.79	1161	596	79072	79063	0.01%
32-30-200-300	46440	132.54	46435	497.74	9.11	3602.80	1241	661	46440	46435	0.01%	3601.67	1217	718	46437	46435	0.00%
32-30-230-300	23500	130.23	23497	621.84	10.83	3603.35	1093	562	23500	23497	0.01%	3599.94	999	594	23500	23497	0.01%
32-30-250-300	6280	154.61	6279	730.32	17.02	3600.02	939	503	6280	6279	0.02%	3604.01	845	487	6280	6279	0.02%
33-30-200-300	55926	110.64	55911	456.08	9.86	17.80	3	3	55911	55911	0.00%	3602.16	1157	639	55917	55911	0.01%
33-30-230-300	48200.5	268.12	48194	933.43	11.13	3600.72	1051	535	48200.5	48194	0.01%	72.73	15	15	48194	48194	0.00%
33-30-250-300	33573	101.29	33569	417.62	11.59	3600.71	1029	539	33573	33569	0.01%	3604.22	999	534	33573	33569	0.01%
34-30-200-300	4623.5	131.44	4623	639.94	14.44	3601.69	953	532	4623.5	4623	0.01%	827.30	145	145	4623	4623	0.00%
34-30-230-300	62736.5	164.23	62724	903.13	12.19	3600.89	993	530	62736.5	62724	0.02%	3599.76	1007	503	62736.5	62724	0.02%
34-30-250-300	*	*	*	*	12.77	3603.27	911	473	16987	16985	0.01%	3602.95	855	487	16987	16985	0.01%
35-30-200-300	29115	115.12	29110	536.07	11.06	3602.77	1089	558	29115	29110	0.02%	3601.26	1079	539	29115	29110	0.02%
35-30-230-300	*	*	*	*	13.41	3603.54	923	478	76030	76015	0.02%	3599.52	941	470	76030	76015	0.02%
35-30-250-300	*	*	*	*	13.90	3605.47	879	457	47615.5	47613	0.01%	3606.11	863	431	47615.5	47613	0.01%
36-30-200-300	40294.5	116.12	40288	615.03	13.64	3603.22	985	530	40294.5	40288	0.02%	3600.02	861	635	40294	40288	0.01%
36-30-230-300	*	*	*	*	14.57	3602.22	875	451	52908	52902	0.01%	1125.79	191	191	52902	52902	0.00%
36-30-250-300	*	*	*	*	14.50	3602.71	859	435	62943.5	62932	0.02%	3604.05	813	406	62943.5	62932	0.02%
37-30-200-300	57430	144.08	57422	826.57	13.07	3605.12	887	519	57430	57422	0.01%	3600.67	761	620	57430	57422	0.01%
37-30-230-300	*	*	*	*	16.65	3600.79	777	419	7770.5	7769	0.02%	3604.85	785	414	7770.5	7769	0.02%
37-30-250-300	*	*	*	*	16.07	3600.22	643	341	41585	41580	0.01%	3600.07	749	374	41585	41580	0.01%
38-30-200-300	*	*	*	*	14.08	3599.36	861	475	28809.5	28805	0.02%	2055.45	351	351	28805	28805	0.00%
38-30-230-300	*	*	*	*	15.72	3599.01	743	443	73705.5	73698	0.01%	3600.67	781	407	73705.5	73698	0.01%
38-30-250-300	*	*	*	*	18.31	3603.98	691	364	24711	24709	0.01%	3601.59	731	367	24711	24709	0.01%
39-30-200-300	*	*	*	*	18.71	3605.31	625	373	4989.5	4989	0.01%	3603.48	691	375	4989.5	4989	0.01%
39-30-230-300	*	*	*	*	17.87	3603.59	671	346	67965	67958	0.01%	2312.17	287	287	67958	67958	0.00%
39-30-250-300	*	*	*	*	17.57	3602.40	693	366	63911.5	63894	0.03%	3600.56	717	358	63911.5	63894	0.03%

Table B.8: Results of the large-scale instances (number of terminals in the range [30,39]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
30-30-200-300	14513.5	100.44	14512	392.92	8.23	18.33	3	3	14512	14512	0.00%	16.41	3	3	14512	14512	0.00%
30-30-230-300	47060.5	104.13	47054	435.42	9.41	24.20	5	5	47054	47054	0.00%	21.33	5	5	47054	47054	0.00%
30-30-250-300	69007	153.42	68998	750.87	10.03	122.18	29	29	68998	68998	0.00%	108.11	29	29	68998	68998	0.00%
31-30-200-300	51227.5	98.37	51221	436.45	8.46	12.80	3	3	51221	51221	0.00%	11.79	3	3	51221	51221	0.00%
31-30-230-300	79462.5	135.00	79441	716.02	9.68	1732.76	431	431	79441	79441	0.00%	1539.31	431	431	79441	79441	0.00%
31-30-250-300	79072	139.71	79063	685.50	10.34	14.41	3	3	79063	79063	0.00%	13.13	3	3	79063	79063	0.00%
32-30-200-300	46440	132.54	46435	497.74	9.11	43.18	11	11	46435	46435	0.00%	39.08	11	11	46435	46435	0.00%
32-30-230-300	23500	130.23	23497	621.84	10.83	19.39	3	3	23497	23497	0.00%	18.04	3	3	23497	23497	0.00%
32-30-250-300	6280	154.61	6279	730.32	17.02	438.41	65	65	6279	6279	0.00%	394.39	65	65	6279	6279	0.00%
33-30-200-300	55926	110.64	55911	456.08	9.86	285.58	63	63	55911	55911	0.00%	254.35	63	63	55911	55911	0.00%
33-30-230-300	48200.5	268.12	48194	933.43	11.13	36.99	7	7	48194	48194	0.00%	32.99	7	7	48194	48194	0.00%
33-30-250-300	33573	101.29	33569	417.62	11.59	73.21	15	15	33569	33569	0.00%	65.80	15	15	33569	33569	0.00%
34-30-200-300	4623.5	131.44	4623	639.94	14.44	69.21	11	11	4623	4623	0.00%	63.03	11	11	4623	4623	0.00%
34-30-230-300	62736.5	164.23	62724	903.13	12.19	62.82	11	11	62724	62724	0.00%	56.14	11	11	62724	62724	0.00%
34-30-250-300	*	*	*	*	12.77	24.03	3	3	16985	16985	0.00%	21.54	3	3	16985	16985	0.00%
35-30-200-300	29115	115.12	29110	536.07	11.06	188.37	39	39	29110	29110	0.00%	170.60	39	39	29110	29110	0.00%
35-30-230-300	*	*	*	*	13.41	50.64	7	7	76015	76015	0.00%	45.84	7	7	76015	76015	0.00%
35-30-250-300	*	*	*	*	13.90	67.28	11	11	47613	47613	0.00%	61.65	11	11	47613	47613	0.00%
36-30-200-300	40294.5	116.12	40288	615.03	13.64	32.82	5	5	40288	40288	0.00%	29.92	5	5	40288	40288	0.00%
36-30-230-300	*	*	*	*	14.57	31.18	5	5	52902	52902	0.00%	28.51	5	5	52902	52902	0.00%
36-30-250-300	*	*	*	*	14.50	24.01	3	3	62932	62932	0.00%	21.88	3	3	62932	62932	0.00%
37-30-200-300	57430	144.08	57422	826.57	13.07	18.33	3	3	57422	57422	0.00%	16.96	3	3	57422	57422	0.00%
37-30-230-300	*	*	*	*	16.65	1375.93	197	197	7769	7769	0.00%	1250.38	197	197	7769	7769	0.00%
37-30-250-300	*	*	*	*	16.07	52.34	7	7	41580	41580	0.00%	48.15	7	7	41580	41580	0.00%
38-30-200-300	*	*	*	*	14.08	513.11	85	85	28805	28805	0.00%	465.01	85	85	28805	28805	0.00%
38-30-230-300	*	*	*	*	15.72	27.76	3	3	73698	73698	0.00%	25.18	3	3	73698	73698	0.00%
38-30-250-300	*	*	*	*	18.31	29.58	3	3	24709	24709	0.00%	26.90	3	3	24709	24709	0.00%
39-30-200-300	*	*	*	*	18.71	74.92	7	7	4989	4989	0.00%	69.13	7	7	4989	4989	0.00%
39-30-230-300	*	*	*	*	17.87	261.91	33	33	67958	67958	0.00%	238.48	33	33	67958	67958	0.00%
39-30-250-300	*	*	*	*	17.57	67.42	9	9	63894	63894	0.00%	60.95	9	9	63894	63894	0.00%

Table B.9: Results of the large-scale instances (number of terminals in the range [40,49]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
40-36-130-500	48491	233.74	48485	1354.03	15.23	3602.48	717	538	48491	48485	0.01%	3237.08	479	479	48485	48485	0.00%
40-36-150-500	*	*	*	*	17.29	67.74	7	7	28510	28510	0.00%	3599.59	571	495	28522	28510	0.04%
40-36-170-500	*	*	*	*	18.22	3601.70	715	384	76115	76106	0.01%	3599.25	587	467	76112	76106	0.01%
41-36-130-500	4350	409.28	4349	2428.86	18.11	3599.87	681	372	4350	4349	0.02%	181.75	19	19	4349	4349	0.00%
41-36-150-500	*	*	*	*	19.53	3599.82	689	424	63709.5	63691	0.03%	3600.84	533	528	63709.5	63691	0.03%
41-36-170-500	*	*	*	*	20.81	3605.97	661	386	54557.5	54550	0.01%	3604.69	719	359	54557.5	54550	0.01%
42-36-130-500	*	*	*	*	19.18	3600.92	657	459	15600.5	15598	0.02%	316.33	47	47	15598	15598	0.00%
42-36-150-500	*	*	*	*	22.40	3599.92	617	343	8179	8178	0.01%	3599.98	515	343	8179	8178	0.01%
42-36-170-500	*	*	*	*	21.62	3599.28	643	352	36383	36376	0.02%	3606.24	689	352	36383	36376	0.02%
43-36-130-500	*	*	*	*	20.24	3601.13	699	405	35992	35988	0.01%	3600.71	749	398	35992	35988	0.01%
43-36-150-500	*	*	*	*	21.36	3603.01	649	356	53283.5	53269	0.03%	3605.25	717	364	53283.5	53269	0.03%
43-36-170-500	*	*	*	*	21.77	3600.57	525	407	37485	37484	0.00%	605.48	71	71	37484	37484	0.00%
44-36-130-500	*	*	*	*	20.28	3599.12	655	456	55765	55748	0.03%	3605.60	527	473	55764	55748	0.03%
44-36-150-500	*	*	*	*	22.88	3601.16	703	394	70091.5	70064	0.04%	630.51	71	71	70064	70064	0.00%
44-36-170-500	*	*	*	*	25.29	3600.28	601	322	37252	37247	0.01%	764.53	75	75	37247	37247	0.00%
45-36-130-500	*	*	*	*	21.41	162.08	23	23	35017	35017	0.00%	132.62	17	17	35017	35017	0.00%
45-36-150-500	*	*	*	*	22.82	3600.60	635	377	35098	35095	0.01%	3603.06	611	353	35098	35095	0.01%
45-36-170-500	*	*	*	*	25.23	3604.34	583	343	67077.5	67070	0.01%	909.36	95	95	67070	67070	0.00%
46-36-130-500	*	*	*	*	26.07	3601.45	547	331	4479.5	4479	0.01%	3601.70	439	312	4479.5	4479	0.01%
46-36-150-500	*	*	*	*	24.76	36.31	3	3	25481	25481	0.00%	3603.24	599	308	25487	25481	0.02%
46-36-170-500	*	*	*	*	28.27	3606.56	573	333	45356.5	45355	0.00%	3600.25	579	289	45356.5	45349	0.02%
47-36-130-500	*	*	*	*	34.85	3605.51	521	303	4825.5	4825	0.01%	3603.99	375	288	4825.5	4825	0.01%
47-36-150-500	*	*	*	*	26.57	3599.70	579	341	54144	54139	0.01%	3603.90	527	315	54140	54139	0.00%
47-36-170-500	*	*	*	*	30.42	3605.26	547	315	46159.5	46154	0.01%	189.65	15	15	46154	46154	0.00%
48-36-130-500	*	*	*	*	28.28	3603.58	553	338	26074	26069	0.02%	3604.40	493	296	26074	26069	0.02%
48-36-150-500	*	*	*	*	33.14	3600.04	481	274	4848.5	4848	0.01%	3606.65	409	253	4848.5	4848	0.01%
48-36-170-500	*	*	*	*	29.44	3602.53	519	293	68487.5	68476	0.02%	3600.23	513	256	68487.5	68476	0.02%
49-36-130-500	*	*	*	*	28.81	3602.83	545	389	38307	38305	0.01%	3599.53	569	327	38307	38304	0.01%
49-36-150-500	*	*	*	*	40.63	3600.85	441	259	4944.5	4944	0.01%	3607.31	433	244	4944.5	4944	0.01%
49-36-170-500	*	*	*	*	34.51	93.03	7	7	4676	4676	0.00%	72.35	5	5	4676	4676	0.00%

Table B.10: Results of the large-scale instances (number of terminals in the range [40,49]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
40-36-130-500	48491	233.74	48485	1354.03	15.23	51.52	7	7	48485	48485	0.00%	47.90	7	7	48485	48485	0.00%
40-36-150-500	*	*	*	*	17.29	35.14	5	5	28510	28510	0.00%	32.65	5	5	28510	28510	0.00%
40-36-170-500	*	*	*	*	18.22	56.59	7	7	76106	76106	0.00%	52.27	7	7	76106	76106	0.00%
41-36-130-500	4350	409.28	4349	2428.86	18.11	123.29	13	13	4349	4349	0.00%	113.69	13	13	4349	4349	0.00%
41-36-150-500	*	*	*	*	19.53	25.77	3	3	63691	63691	0.00%	24.16	3	3	63691	63691	0.00%
41-36-170-500	*	*	*	*	20.81	26.75	3	3	54550	54550	0.00%	25.21	3	3	54550	54550	0.00%
42-36-130-500	*	*	*	*	19.18	43.52	5	5	15598	15598	0.00%	42.48	5	5	15598	15598	0.00%
42-36-150-500	*	*	*	*	22.40	27.73	3	3	8178	8178	0.00%	26.89	3	3	8178	8178	0.00%
42-36-170-500	*	*	*	*	21.62	86.85	11	11	36376	36376	0.00%	80.60	11	11	36376	36376	0.00%
43-36-130-500	*	*	*	*	20.24	26.83	3	3	35988	35988	0.00%	25.25	3	3	35988	35988	0.00%
43-36-150-500	*	*	*	*	21.36	152.43	17	17	53269	53269	0.00%	138.62	17	17	53269	53269	0.00%
43-36-170-500	*	*	*	*	21.77	70.46	7	7	37484	37484	0.00%	65.34	7	7	37484	37484	0.00%
44-36-130-500	*	*	*	*	20.28	87.39	11	11	55748	55748	0.00%	79.92	11	11	55748	55748	0.00%
44-36-150-500	*	*	*	*	22.88	43.50	3	3	70064	70064	0.00%	39.81	3	3	70064	70064	0.00%
44-36-170-500	*	*	*	*	25.29	141.57	11	11	37247	37247	0.00%	128.38	11	11	37247	37247	0.00%
45-36-130-500	*	*	*	*	21.41	40.88	5	5	35017	35017	0.00%	38.40	5	5	35017	35017	0.00%
45-36-150-500	*	*	*	*	22.82	73.93	7	7	35095	35095	0.00%	69.02	7	7	35095	35095	0.00%
45-36-170-500	*	*	*	*	25.23	38.66	3	3	67070	67070	0.00%	35.68	3	3	67070	67070	0.00%
46-36-130-500	*	*	*	*	26.07	425.27	35	35	4479	4479	0.00%	393.58	35	35	4479	4479	0.00%
46-36-150-500	*	*	*	*	24.76	251.81	27	27	25481	25481	0.00%	232.55	27	27	25481	25481	0.00%
46-36-170-500	*	*	*	*	28.27	40.57	3	3	45355	45355	0.00%	37.83	3	3	45355	45355	0.00%
47-36-130-500	*	*	*	*	34.85	74.42	5	5	4825	4825	0.00%	69.29	5	5	4825	4825	0.00%
47-36-150-500	*	*	*	*	26.57	81.61	7	7	54139	54139	0.00%	76.43	7	7	54139	54139	0.00%
47-36-170-500	*	*	*	*	30.42	84.69	7	7	46154	46154	0.00%	78.83	7	7	46154	46154	0.00%
48-36-130-500	*	*	*	*	28.28	98.27	9	9	26069	26069	0.00%	91.27	9	9	26069	26069	0.00%
48-36-150-500	*	*	*	*	33.14	127.07	9	9	4848	4848	0.00%	118.50	9	9	4848	4848	0.00%
48-36-170-500	*	*	*	*	29.44	178.74	15	15	68476	68476	0.00%	163.78	15	15	68476	68476	0.00%
49-36-130-500	*	*	*	*	28.81	40.23	3	3	38305	38305	0.00%	37.78	3	3	38305	38305	0.00%
49-36-150-500	*	*	*	*	40.63	130.17	9	9	4944	4944	0.00%	121.98	9	9	4944	4944	0.00%
49-36-170-500	*	*	*	*	34.51	76.62	5	5	4676	4676	0.00%	72.37	5	5	4676	4676	0.00%

Table B.11: Results of the large-scale instances (number of terminals in the range [50,54]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
50-36-100-700	*	*	*	*	23.56	3599.42	527	424	50963.5	50961	0.00%	36.77	3	3	50961	50961	0.00%
50-36-130-700	*	*	*	*	29.24	3600.10	563	316	57063.5	57048	0.03%	3600.59	375	333	57063.5	57048	0.03%
50-36-150-700	*	*	*	*	31.62	40.88	3	3	22655	22655	0.00%	3602.52	477	264	22658.5	22656	0.01%
50-36-180-700	*	*	*	*	34.10	3608.23	469	246	62142	62122	0.03%	3600.63	451	264	62141.5	62122	0.03%
50-36-200-700	*	*	*	*	34.85	3609.31	441	229	28848	28843	0.02%	3607.46	421	242	28848	28843	0.02%
50-36-250-700	*	*	*	*	40.64	56.93	3	3	111444	111444	0.00%	3613.44	367	183	111459	111444	0.01%
51-36-100-700	*	*	*	*	24.03	3356.60	399	399	17676	17676	0.00%	62.19	7	7	17676	17676	0.00%
51-36-130-700	*	*	*	*	30.62	3606.02	545	284	62850	62845	0.01%	2502.91	223	223	62845	62845	0.00%
51-36-150-700	*	*	*	*	30.51	3605.89	497	274	61616	61608	0.01%	3599.87	473	286	61616	61608	0.01%
51-36-180-700	*	*	*	*	37.18	3604.87	423	250	45128.5	45116	0.03%	3609.76	429	229	45128.5	45116	0.03%
51-36-200-700	*	*	*	*	38.92	3605.90	407	217	69131.5	69124	0.01%	3609.21	327	255	69127	69124	0.00%
51-36-250-700	*	*	*	*	49.22	3613.18	319	176	29395.5	29392	0.01%	3602.35	327	172	29395.5	29392	0.01%
52-36-100-700	*	*	*	*	25.76	231.81	27	27	47273	47273	0.00%	79.83	9	9	47273	47273	0.00%
52-36-130-700	*	*	*	*	33.24	3600.56	515	287	60092.5	60081	0.02%	3607.32	551	275	60092	60081	0.02%
52-36-150-700	*	*	*	*	28.20	3607.79	447	273	75999	75988	0.01%	3608.18	355	305	75999	75988	0.01%
52-36-180-700	*	*	*	*	37.01	3609.97	415	225	54700.5	54693	0.01%	3606.44	387	249	54700.5	54693	0.01%
52-36-200-700	*	*	*	*	43.32	3610.06	355	190	33918	33908	0.03%	3610.95	363	194	33918	33908	0.03%
52-36-250-700	*	*	*	*	52.17	3614.09	283	153	8242	8241	0.01%	3604.14	291	154	8242	8241	0.01%
53-36-100-700	*	*	*	*	26.39	1775.47	199	199	18727	18727	0.00%	218.01	23	23	18727	18727	0.00%
53-36-130-700	*	*	*	*	31.03	3600.58	469	297	36395.5	36388	0.02%	3604.30	521	286	36393.5	36388	0.02%
53-36-150-700	*	*	*	*	38.67	3606.57	437	251	63064.5	63060	0.01%	454.83	39	39	63060	63060	0.00%
53-36-180-700	*	*	*	*	38.71	3601.22	395	208	28789	28785	0.01%	3602.49	401	213	28787.5	28785	0.01%
53-36-220-700	*	*	*	*	42.52	3609.63	345	201	101741	101729	0.01%	3612.15	365	182	101741	101723	0.02%
53-36-250-700	*	*	*	*	49.47	3601.82	303	179	104928	104922	0.01%	65.66	3	3	104922	104922	0.00%
54-36-100-700	*	*	*	*	29.48	3606.00	573	301	23558	23551	0.03%	999.09	95	95	23551	23551	0.00%
54-36-130-700	*	*	*	*	33.05	3602.89	349	293	39538.5	39535	0.01%	667.68	61	61	39535	39535	0.00%
54-36-150-700	*	*	*	*	38.63	3601.22	349	216	4722.5	4722	0.01%	3605.55	277	213	4722.5	4722	0.01%
54-36-180-700	*	*	*	*	40.40	3601.80	297	208	36089	36084	0.01%	57.98	3	3	36084	36084	0.00%
54-36-200-700	*	*	*	*	41.92	3600.62	355	195	54219	54198	0.04%	3601.31	333	210	54219	54198	0.04%
54-36-250-700	*	*	*	*	57.90	3615.92	273	144	8532	8531	0.01%	3599.89	213	149	8532	8531	0.01%

Table B.12: Results of the large-scale instances (number of terminals in the range [50,54]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
50-36-100-700	*	*	*	*	23.56	36.61	3	3	50961	50961	0.00%	34.71	3	3	50961	50961	0.00%
50-36-130-700	*	*	*	*	29.24	3606.93	495	306	-57052.5	57048	0.01%	3601.84	525	328	57052.5	57048	0.01%
50-36-150-700	*	*	*	*	31.62	68.46	5	5	22656	22656	0.00%	68.35	5	5	22656	22656	0.00%
50-36-180-700	*	*	*	*	34.10	630.27	57	57	62122	62122	0.00%	627.10	57	57	62122	62122	0.00%
50-36-200-700	*	*	*	*	34.85	38.00	3	3	28843	28843	0.00%	37.93	3	3	28843	28843	0.00%
50-36-250-700	*	*	*	*	40.64	3600.66	329	213	111450	111444	0.01%	3611.77	329	216	111450	111444	0.01%
51-36-100-700	*	*	*	*	24.03	99.38	11	11	17676	17676	0.00%	99.31	11	11	17676	17676	0.00%
51-36-130-700	*	*	*	*	30.62	42.16	3	3	62845	62845	0.00%	42.11	3	3	62845	62845	0.00%
51-36-150-700	*	*	*	*	30.51	171.83	17	17	61608	61608	0.00%	171.96	17	17	61608	61608	0.00%
51-36-180-700	*	*	*	*	37.18	1802.81	153	153	45116	45116	0.00%	1798.33	153	153	45116	45116	0.00%
51-36-200-700	*	*	*	*	38.92	54.47	3	3	69124	69124	0.00%	55.16	3	3	69124	69124	0.00%
51-36-250-700	*	*	*	*	49.22	767.03	43	43	29392	29392	0.00%	772.57	43	43	29392	29392	0.00%
52-36-100-700	*	*	*	*	25.76	38.55	3	3	47273	47273	0.00%	38.97	3	3	47273	47273	0.00%
52-36-130-700	*	*	*	*	33.24	62.32	5	5	60081	60081	0.00%	63.88	5	5	60081	60081	0.00%
52-36-150-700	*	*	*	*	28.20	45.83	3	3	75988	75988	0.00%	46.27	3	3	75988	75988	0.00%
52-36-180-700	*	*	*	*	37.01	206.15	17	17	54693	54693	0.00%	208.32	17	17	54693	54693	0.00%
52-36-200-700	*	*	*	*	43.32	89.17	5	5	33908	33908	0.00%	91.11	5	5	33908	33908	0.00%
52-36-250-700	*	*	*	*	52.17	223.27	13	9	8242	8242	0.00%	224.81	13	9	8242	8242	0.00%
53-36-100-700	*	*	*	*	26.39	147.35	15	15	18727	18727	0.00%	150.27	15	15	18727	18727	0.00%
53-36-130-700	*	*	*	*	31.03	51.23	3	3	36388	36388	0.00%	51.06	3	3	36388	36388	0.00%
53-36-150-700	*	*	*	*	38.67	54.46	3	3	63060	63060	0.00%	55.33	3	3	63060	63060	0.00%
53-36-180-700	*	*	*	*	38.71	1020.99	77	77	28785	28785	0.00%	1023.86	77	77	28785	28785	0.00%
53-36-220-700	*	*	*	*	42.52	106.86	7	7	101729	101729	0.00%	105.90	7	7	101729	101729	0.00%
53-36-250-700	*	*	*	*	49.47	65.29	3	3	104922	104922	0.00%	65.19	3	3	104922	104922	0.00%
54-36-100-700	*	*	*	*	29.48	135.83	13	13	23551	23551	0.00%	137.10	13	13	23551	23551	0.00%
54-36-130-700	*	*	*	*	33.05	45.49	3	3	39535	39535	0.00%	45.46	3	3	39535	39535	0.00%
54-36-150-700	*	*	*	*	38.63	79.96	5	5	4722	4722	0.00%	81.05	5	5	4722	4722	0.00%
54-36-180-700	*	*	*	*	40.40	107.57	7	7	36084	36084	0.00%	107.63	7	7	36084	36084	0.00%
54-36-200-700	*	*	*	*	41.92	44.16	3	3	54198	54198	0.00%	45.06	3	3	54198	54198	0.00%
54-36-250-700	*	*	*	*	57.90	1416.50	59	59	8531	8531	0.00%	1418.64	59	59	8531	8531	0.00%

Table B.13: Results of the large-scale instances (number of terminals in the range [55,59]) for testing the B&P with schemes A and B

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-A	N.T-A	N.E-A	UB BP-A	LB BP-A	GAP-A	CPU BP-B	N.T-B	N.E-B	UB BP-B	LB BP-B	GAP-B
55-36-100-700	*	*	*	*	29.75	3600.72	393	322	28198	28195	0.01%	3600.97	383	338	28198	28195	0.01%
55-36-130-700	*	*	*	*	47.77	3602.89	381	210	4506.5	4506	0.01%	3600.54	375	215	4506.5	4506	0.01%
55-36-150-700	*	*	*	*	40.81	3609.54	259	203	7854	7853	0.01%	3599.16	313	203	7854	7853	0.01%
55-36-180-700	*	*	*	*	43.37	3601.29	381	214	40858	40857	0.00%	67.67	5	3	40857	40857	0.00%
55-36-200-700	*	*	*	*	45.23	3599.78	257	226	29967.5	29966	0.01%	3606.88	315	197	29971	29966	0.02%
55-36-250-700	*	*	*	*	55.72	3600.92	289	152	54663	54658	0.01%	3614.81	291	146	54663	54657	0.01%
56-36-100-700	*	*	*	*	35.63	3601.71	451	248	3481.5	3481	0.01%	205.45	13	13	3481	3481	0.00%
56-36-130-700	*	*	*	*	52.44	3600.99	355	211	4433.5	4433	0.01%	61.65	3	3	4433	4433	0.00%
56-36-150-700	*	*	*	*	41.97	3603.64	299	207	11530	11529	0.01%	972.77	61	61	11529	11529	0.00%
56-36-180-700	*	*	*	*	48.28	3600.51	341	196	44020.5	44014	0.01%	3601.07	247	202	44023.5	44014	0.02%
56-36-200-700	*	*	*	*	51.13	3613.55	333	176	71442	71438	0.01%	207.12	11	11	71438	71438	0.00%
56-36-250-700	*	*	*	*	54.70	3614.35	283	143	85918.5	85911	0.01%	206.20	11	11	85911	85911	0.00%
57-36-100-700	*	*	*	*	33.38	327.81	23	23	12646	12646	0.00%	77.18	5	5	12646	12646	0.00%
57-36-130-700	*	*	*	*	37.13	3600.76	431	242	59524	59518	0.01%	908.61	77	77	59518	59518	0.00%
57-36-150-700	*	*	*	*	40.73	3604.38	347	239	51994	51978	0.03%	83.10	5	5	51978	51978	0.00%
57-36-180-700	*	*	*	*	47.38	3600.93	339	193	62349	62345	0.01%	3612.15	327	196	62349	62345	0.01%
57-36-200-700	*	*	*	*	52.76	3602.41	317	162	91817	91807	0.01%	3601.25	309	162	91816	91807	0.01%
57-36-250-700	*	*	*	*	62.00	3609.53	247	137	47469	47464	0.01%	3614.37	229	157	47469	47464	0.01%
58-36-100-700	*	*	*	*	48.58	186.78	15	15	31050	31050	0.00%	46.07	3	3	31050	31050	0.00%
58-36-130-700	*	*	*	*	31.85	98.67	7	7	18790	18790	0.00%	3606.96	391	226	18794.5	18790	0.02%
58-36-150-700	*	*	*	*	53.58	3611.26	367	197	64731	64722	0.01%	3605.91	391	195	64729.5	64722	0.01%
58-36-180-700	*	*	*	*	42.30	3609.26	321	162	49236	49225	0.02%	3610.68	261	202	49234.5	49225	0.02%
58-36-200-700	*	*	*	*	62.03	3606.82	287	160	24634.5	24632	0.01%	872.66	45	45	24632	24632	0.00%
58-36-250-700	*	*	*	*	46.64	3613.72	251	132	116441	116424	0.01%	3600.33	203	163	116441	116424	0.01%
59-36-100-700	*	*	*	*	46.65	3600.46	391	210	3566.5	3566	0.01%	3600.35	257	245	3566.5	3566	0.01%
59-36-130-700	*	*	*	*	50.34	3609.76	343	182	4957	4956	0.02%	3601.53	321	180	4957	4956	0.02%
59-36-150-700	*	*	*	*	60.56	3600.52	287	172	5375	5374	0.02%	202.02	9	9	5374	5374	0.00%
59-36-180-700	*	*	*	*	52.15	3606.49	317	178	83515.5	83504	0.01%	3602.34	335	167	83515.5	83504	0.01%
59-36-200-700	*	*	*	*	55.35	3601.03	291	163	46813	46808	0.01%	3602.20	261	174	46813	46808	0.01%
59-36-250-700	*	*	*	*	63.43	290.80	13	11	96410	96410	0.00%	1802.07	81	81	96410	96410	0.00%

Table B.14: Results of the large-scale instances (number of terminals in the range [55,59]) for testing the B&P with schemes C and D

Instance	LP	CPU LP	IP	CPU IP	CPU PD	CPU BP-C	N.T-C	N.E-C	UB BP-C	LB BP-C	GAP-C	CPU BP-D	N.T-D	N.E-D	UB BP-D	LB BP-D	GAP-D
55-36-100-700	*	*	*	*	29.75	99.24	9	9	28195	28195	0.00%	100.46	9	9	28195	28195	0.00%
55-36-130-700	*	*	*	*	47.77	73.85	3	3	4506	4506	0.00%	73.97	3	3	4506	4506	0.00%
55-36-150-700	*	*	*	*	40.81	49.21	3	3	7853	7853	0.00%	49.44	3	3	7853	7853	0.00%
55-36-180-700	*	*	*	*	43.37	108.91	7	7	40857	40857	0.00%	109.13	7	7	40857	40857	0.00%
55-36-200-700	*	*	*	*	45.23	120.64	7	7	29966	29966	0.00%	120.55	7	7	29966	29966	0.00%
55-36-250-700	*	*	*	*	55.72	108.39	5	5	54658	54658	0.00%	108.08	5	5	54658	54658	0.00%
56-36-100-700	*	*	*	*	35.63	54.02	3	3	3481	3481	0.00%	54.07	3	3	3481	3481	0.00%
56-36-130-700	*	*	*	*	52.44	57.47	3	3	4433	4433	0.00%	57.63	3	3	4433	4433	0.00%
56-36-150-700	*	*	*	*	41.97	97.21	5	5	11529	11529	0.00%	98.02	5	5	11529	11529	0.00%
56-36-180-700	*	*	*	*	48.28	244.52	17	15	44014	44014	0.00%	244.86	17	15	44014	44014	0.00%
56-36-200-700	*	*	*	*	51.13	65.49	3	3	71438	71438	0.00%	65.10	3	3	71438	71438	0.00%
56-36-250-700	*	*	*	*	54.70	1929.39	111	101	85911	85911	0.00%	1942.71	111	101	85911	85911	0.00%
57-36-100-700	*	*	*	*	33.38	49.64	3	3	12646	12646	0.00%	49.92	3	3	12646	12646	0.00%
57-36-130-700	*	*	*	*	37.13	150.45	11	11	59518	59518	0.00%	153.09	11	11	59518	59518	0.00%
57-36-150-700	*	*	*	*	40.73	80.02	5	5	51978	51978	0.00%	80.27	5	5	51978	51978	0.00%
57-36-180-700	*	*	*	*	47.38	62.45	3	3	62345	62345	0.00%	62.52	3	3	62345	62345	0.00%
57-36-200-700	*	*	*	*	52.76	57.51	3	3	91807	91807	0.00%	57.93	3	3	91807	91807	0.00%
57-36-250-700	*	*	*	*	62.00	63.95	3	3	47464	47464	0.00%	63.98	3	3	47464	47464	0.00%
58-36-100-700	*	*	*	*	48.58	99.01	7	7	31050	31050	0.00%	99.66	7	7	31050	31050	0.00%
58-36-130-700	*	*	*	*	31.85	48.85	3	3	18790	18790	0.00%	49.34	3	3	18790	18790	0.00%
58-36-150-700	*	*	*	*	53.58	51.33	3	3	64722	64722	0.00%	51.38	3	3	64722	64722	0.00%
58-36-180-700	*	*	*	*	42.30	161.57	9	9	49225	49225	0.00%	164.57	9	9	49225	49225	0.00%
58-36-200-700	*	*	*	*	62.03	79.92	3	3	24632	24632	0.00%	80.03	3	3	24632	24632	0.00%
58-36-250-700	*	*	*	*	46.64	2272.57	113	113	116424	116424	0.00%	2271.38	113	113	116424	116424	0.00%
59-36-100-700	*	*	*	*	46.65	515.08	35	35	3566	3566	0.00%	511.11	35	35	3566	3566	0.00%
59-36-130-700	*	*	*	*	50.34	1140.72	61	61	4956	4956	0.00%	1147.60	61	61	4956	4956	0.00%
59-36-150-700	*	*	*	*	60.56	79.61	3	3	5374	5374	0.00%	79.93	3	3	5374	5374	0.00%
59-36-180-700	*	*	*	*	52.15	255.74	15	15	83504	83504	0.00%	256.82	15	15	83504	83504	0.00%
59-36-200-700	*	*	*	*	55.35	75.13	3	3	46808	46808	0.00%	73.87	3	3	46808	46808	0.00%
59-36-250-700	*	*	*	*	63.43	82.43	3	3	96410	96410	0.00%	82.01	3	3	96410	96410	0.00%

Appendix C

Results of node-demand formulation and Branch-and-Price

This appendix shows the detailed results of the computational experiments described in Section 6.4, using CPLEX to solve the node-demand and arc-demand models and the branch-and-price (BP) methods based on these models. Recall that a time limit of 7200 seconds was imposed on each run. The tables in this appendix follow a similar structure as in Section 6.4. For each instance and solution approach, they may show the following columns:

- *Instance* is the name of the instance.
- *LP OV* is the optimal value of the LP relaxation of the compact model.
- *IP OV* is the optimal value of the compact model.
- *LP* is the time taken by CPLEX to solve the LP relaxation of the compact model.
- *IP* is the time taken by CPLEX to solve the compact model.
- T_{Root} (sec) is the time taken by the BP method to solve the root node only.
- *UB* is the best upper bound reached by the BP method.
- ILP_{OV} is the best lower bound reached by the BP method (the optimal value if the method finishes before the time limit).
- *Gap* is the relative optimality gap between the values of *UB* and ILP_{OV} reached the BP method.

All times are displayed in seconds. A sign "*" indicates that CPLEX could not solve or even mount the model due to lack of computer memory.

Table C.1: Results from solving the compact model of the arc-based and node-based formulations in instances with terminals ranging from 10 to 14.

Instance	*LP OV*	*IP OV*	*Arc-demand*		*Node-demand*	
			LP	*ILP*	*LP*	*ILP*
10-10-20-20	1940	1937	0.13	0.52	0.00	0.03
10-10-20-25	1874	1857	0.06	0.12	0.01	0.02
10-10-20-30	507	501	0.05	0.10	0.00	0.01
10-10-20-35	2474.5	2464	0.05	0.16	0.01	0.02
10-10-20-50	422.5	421	0.07	0.15	0.02	0.10
11-10-20-20	1601	1584	0.06	0.50	0.00	0.01
11-10-20-25	661.5	659	0.07	0.13	0.00	0.40
11-10-20-30	1606.5	1596	0.06	0.16	0.00	0.01
11-10-20-35	2240.5	2238	0.06	0.29	0.01	0.02
11-10-20-50	2704.5	2701	0.08	0.26	0.03	0.03
12-10-20-20	907	904	0.08	0.59	0.00	0.07
12-10-20-25	396.5	396	0.08	0.21	0.00	0.02
12-10-20-30	337	335	0.08	0.17	0.00	0.19
12-10-20-35	677.5	669	0.08	0.18	0.01	0.02
12-10-20-50	811.5	811	0.08	0.16	0.02	0.02
13-10-20-20	1259.5	1259	0.10	0.18	0.00	0.02
13-10-20-25	671.5	669	0.09	0.26	0.00	0.02
13-10-20-30	581	580	0.09	0.17	0.00	0.02
13-10-20-35	267.5	266	0.09	0.16	0.00	0.01
13-10-20-50	680	678	0.09	0.33	0.02	0.02
14-10-20-20	636.5	633	0.11	0.21	0.00	0.03
14-10-20-25	1295	1292	0.12	0.36	0.00	0.04
14-10-20-30	579	574	0.11	0.31	0.01	0.04
14-10-20-35	2058.5	2057	0.11	0.21	0.00	0.01
14-10-20-50	356	355	0.11	0.65	0.02	0.09

Table C.2: Results from solving the compact model of the arc-based and node-based formulations in instances with terminals ranging from 15 to 19.

Instance	*LP OV*	*IP OV*	*Arc-demand*		*Node-demand*	
			LP	*ILP*	*LP*	*ILP*
15-10-20-20	564	560	0.13	0.26	0.00	0.02
15-10-20-25	1202.5	1202	0.13	0.27	0.00	0.03
15-10-20-30	1454.5	1451	0.12	0.26	0.00	0.01
15-10-20-35	1679.5	1679	0.12	0.23	0.01	0.01
15-10-20-50	624	623	0.13	0.36	0.02	0.03
16-10-20-20	738	736	0.14	0.46	0.00	0.02
16-10-20-25	1481.5	1479	0.15	0.29	0.00	0.02
16-10-20-30	1800	1788	0.14	0.28	0.00	0.02
16-10-20-35	2040.5	2037	0.14	0.27	0.00	0.01
16-10-20-50	1564.5	1562	0.15	0.33	0.02	0.03
17-10-20-20	632.5	632	0.16	0.30	0.00	0.01
17-10-20-25	1613.5	1611	0.16	0.31	0.00	0.01
17-10-20-30	424.5	424	0.16	0.28	0.00	0.01
17-10-20-35	2349	2347	0.17	0.32	0.01	0.02
17-10-20-50	2210	2202	0.16	0.37	0.02	0.03
18-10-20-20	1147.5	1143	0.17	0.29	0.00	0.02
18-10-20-25	294.5	291	0.17	0.41	0.00	0.01
18-10-20-30	1836	1835	0.17	0.29	0.01	0.02
18-10-20-35	992	991	0.18	0.78	0.00	0.01
18-10-20-50	550.5	550	0.20	0.35	0.03	0.04
19-10-20-20	906.5	905	0.19	0.33	0.00	0.01
19-10-20-25	535.5	535	0.20	0.36	0.00	0.12
19-10-20-30	1405.5	1402	0.22	0.45	0.00	0.01
19-10-20-35	692.5	691	0.20	0.38	0.00	0.01
19-10-20-50	2599	2591	0.21	0.61	0.02	0.04

Table C.3: Results from solving the compact model of the arc-based and node-based formulations in instances with terminals ranging from 20 to 29.

Instance	*LP OV*	*IP OV*	*Arc-demand*		*Node-demand*	
			LP	*ILP*	*LP*	*ILP*
20-20-100-200	14473	14471	3.49	16.23	1.56	2.64
20-20-130-200	31331	31327	4.33	19.13	1.26	2.26
20-20-150-200	34622.5	34618	5.45	22.10	1.73	2.97
21-20-100-200	2585.5	2585	5.36	19.09	1.58	2.48
21-20-130-200	4056.5	4056	5.01	21.65	1.30	2.27
21-20-150-200	27333.5	27322	5.51	23.34	1.75	2.96
22-20-100-200	4902.5	4902	5.82	23.76	1.46	2.56
22-20-130-200	12304	12301	5.71	25.56	1.88	2.89
22-20-150-200	10972.5	10971	7.14	34.33	2.68	4.20
23-20-100-200	14133.5	14132	4.65	19.48	1.11	1.84
23-20-130-200	21749.5	21745	6.44	27.30	1.62	2.90
23-20-150-200	23366	23358	8.00	31.67	2.15	3.53
24-20-100-200	10157	10155	5.30	24.04	1.33	2.29
24-20-130-200	3486.5	3486	11.08	42.33	2.56	3.51
24-20-150-200	18158	18156	12.68	52.91	2.38	3.87
25-20-100-200	9336.5	9335	5.20	23.05	1.35	2.53
25-20-130-200	33906	33905	8.86	38.92	2.30	4.29
25-20-150-200	32517	32508	9.41	37.72	2.08	2.97
26-20-100-200	9816	9815	5.49	22.13	1.37	2.50
26-20-130-200	16895	16892	7.66	33.17	1.59	2.93
26-20-150-200	3731.5	3731	19.33	67.78	2.39	4.36
27-20-100-200	21095.5	21090	7.05	32.24	1.66	2.38
27-20-130-200	8457.5	8456	11.71	55.14	2.34	3.48
27-20-150-200	26709	26704	9.95	44.40	1.81	2.89
28-20-100-200	3641.5	3641	10.32	42.01	1.89	2.89
28-20-130-200	25359	25355	8.64	36.32	1.44	2.65
28-20-150-200	11225.5	11222	11.36	57.47	2.41	4.07
29-20-100-200	5995	5994	7.09	31.58	1.28	2.15
29-20-130-200	8979.5	8978	11.48	49.86	1.88	2.77
29-20-150-200	7978.5	7977	10.10	44.10	1.88	3.22

Table C.4: Results from solving the compact model of the arc-based and node-based formulations in instances with terminals ranging from 30 to 39.

Instance	*LP OV*	*IP OV*	*Arc-demand*		*Node-demand*	
			LP	*ILP*	*LP*	*ILP*
30-30-200-300	14513.5	14512	100.44	392.92	7.26	9.54
30-30-230-300	47060.5	47054	104.13	435.42	6.84	9.18
30-30-250-300	69007	68998	153.42	750.87	13.72	16.14
31-30-200-300	51227.5	51221	98.37	436.45	7.82	10.46
31-30-230-300	79462.5	79441	135.00	716.02	10.75	13.15
31-30-250-300	79072	79063	139.71	685.50	12.64	14.95
32-30-200-300	46440	46435	132.54	497.74	7.67	9.83
32-30-230-300	23500	23497	130.23	621.84	9.94	11.45
32-30-250-300	6280	6279	154.61	730.32	14.49	16.02
33-30-200-300	55926	55911	110.64	456.08	9.10	10.48
33-30-230-300	48200.5	48194	268.12	933.43	12.97	14.46
33-30-250-300	33573	33569	101.29	417.62	7.85	9.35
34-30-200-300	4623.5	4623	131.44	639.94	10.56	11.93
34-30-230-300	62736.5	62724	164.23	903.13	14.56	16.45
34-30-250-300	16987	*	*	*	12.80	14.17
35-30-200-300	29115	29110	115.12	536.07	7.17	9.46
35-30-230-300	76030	*	*	*	10.95	12.96
35-30-250-300	47615.5	*	*	*	10.39	12.87
36-30-200-300	40294.5	40288	116.12	615.03	8.72	11.07
36-30-230-300	52908	*	*	*	12.44	15.01
36-30-250-300	62943.5	*	*	*	18.74	20.48
37-30-200-300	57430	57422	144.08	826.57	9.75	11.46
37-30-230-300	7770.5	*	*	*	9.46	11.01
37-30-250-300	41585	*	*	*	10.59	12.85
38-30-200-300	28809.5	*	*	*	7.21	9.07
38-30-230-300	73705.5	*	*	*	11.67	13.96
38-30-250-300	24711	*	*	*	12.50	14.76
39-30-200-300	4989.5	*	*	*	8.57	11.01
39-30-230-300	67965	*	*	*	13.52	16.77
39-30-250-300	63911.5	*	*	*	10.49	12.32

Table C.5: Results from solving the compact model of the arc-based and node-based formulations in instances with terminals ranging from 40 to 49.

Instance	*LP OV*	*IP OV*	*Arc-demand*		*Node-demand*	
			LP	*ILP*	*LP*	*ILP*
40-36-130-500	48491	48485	233.74	1354.03	14.75	17.27
40-36-150-500	*	*	*	*	19.29	23.76
40-36-170-500	*	*	*	*	*	*
41-36-130-500	4350	4349	409.28	2428.86	25.71	30.59
41-36-150-500	63709.5	63691	*	*	30.74	32.25
41-36-170-500	*	*	*	*	*	*
42-36-130-500	15601.5	15598	*	*	26.75	29.42
42-36-150-500	8179	8178	*	*	31.74	35.39
42-36-170-500	*	*	*	*	*	*
43-36-130-500	35992	35988	*	*	15.24	19.23
43-36-150-500	53283.5	53269	*	*	22.74	25.60
43-36-170-500	*	*	*	*	*	*
44-36-130-500	55765	55748	*	*	31.96	32.94
44-36-150-500	*	*	*	*	*	*
44-36-170-500	*	*	*	*	*	*
45-36-130-500	35025	35017	*	*	25.74	32.31
45-36-150-500	*	*	*	*	*	*
45-36-170-500	*	*	*	*	*	*
46-36-130-500	4479.5	4479	*	*	24.13	29.08
46-36-150-500		*		*	*	*
46-36-170-500	*	*	*	*	*	*
47-36-130-500	*	*	*	*	*	*
47-36-150-500	*	*	*	*	*	*
47-36-170-500	*	*	*	*	*	*
48-36-130-500	*	*	*	*	*	*
48-36-150-500	*	*	*	*	*	*
48-36-170-500	*	*	*	*	*	*
49-36-130-500	*	*	*	*	*	*
49-36-150-500	*	*	*	*	*	*
49-36-170-500	*	*	*	*	*	*

Table C.6: Results from solving the compact model of the arc-based and node-based formulations in instances from Vasco and Morabito (2016b).

Instance	*LP OV*	*IP OV*	*Arc-demand*		*Node-demand*	
			LP	*ILP*	*LP*	*ILP*
53-36-130-300-p1	17437.8	17437.6	*	*	4.15	28.81
53-36-130-300-p2	19260	19260	*	*	3.19	5.05
53-36-130-300-p3	16634.5	16633.8	*	*	2.34	4.84
53-36-130-300-p4	19560	19560	*	*	4.55	29.11
53-36-130-300-p5	18169.2	18169.2	*	*	2.01	2.75
53-36-130-300-p6	19969.4	19969.4	*	*	5.79	12.30
53-36-130-300-p7	19213.8	19213.8	*	*	2.81	3.42
53-36-130-300-p8	18475.9	18472.6	*	*	4.01	14.68
53-36-130-300-p9	15371.4	15371.4	*	*	3.51	3.85
53-36-130-300-p10	18344.8	18344.8	*	*	3.75	2.64
53-36-130-300-p11	16799.9	16799.6	*	*	2.09	3.39
53-36-130-300-p12	22008.4	22008.4	*	*	4.50	15.54
53-36-130-300-p13	19628.6	19628.2	*	*	3.37	6.87
53-36-130-300-p14	19616.6	19616.6	*	*	2.90	4.34
53-36-130-300-p15	20673.2	20673.2	*	*	4.48	4.04
53-36-130-300-p16	17796.2	17796.2	*	*	3.30	5.05
53-36-130-300-p17	17345.2	17345.2	*	*	2.20	2.24
53-36-130-300-p18	17850.7	17849.2	*	*	3.22	5.99
53-36-130-300-p19	18190.6	18190.6	*	*	2.00	2.23
53-36-130-300-p20	20754.4	20754.4	*	*	2.78	5.51
53-36-130-300-p21	16953.2	16953.2	*	*	7.36	16.13
53-36-130-300-p22	18699.2	18699.2	*	*	5.56	16.86
53-36-130-300-p23	21525.6	21525.6	*	*	4.41	13.97
53-36-130-300-p24	18266.2	18266.2	*	*	2.31	2.60
53-36-130-300-p25	17064.8	17064.8	*	*	3.43	6.48
53-36-130-300-p26	20324.4	20324.4	*	*	2.63	3.50
53-36-130-300-p27	20003	20003	*	*	2.67	2.69
53-36-130-300-p28	17956	17956	*	*	2.79	3.82
53-36-130-300-p29	19074.6	19074.6	*	*	2.21	2.29
53-36-130-300-p30	16464.6	16464.6	*	*	2.69	3.63

Table C.7: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 10 to 14.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
10-10-20-20	0.09	1937	1937	0.0%	0.18	0.006	1937	1937	0.0%	0.06
10-10-20-25	0.06	1857	1857	0.0%	0.15	0.005	1857	1857	0.0%	0.06
10-10-20-30	0.06	501	501	0.0%	0.44	0.005	501	501	0.0%	0.09
10-10-20-35	0.07	2464	2464	0.0%	0.18	0.005	2464	2464	0.0%	0.06
10-10-20-50	0.07	421	421	0.0%	0.15	0.007	421	421	0.0%	0.05
11-10-20-20	0.09	1584	1584	0.0%	0.19	0.005	1584	1584	0.0%	0.06
11-10-20-25	0.09	659	659	0.0%	0.18	0.005	659	659	0.0%	0.03
11-10-20-30	0.06	1596	1596	0.0%	0.18	0.005	1596	1596	0.0%	0.02
11-10-20-35	0.07	2238	2238	0.0%	0.20	0.006	2238	2238	0.0%	0.07
11-10-20-50	0.37	2701	2701	0.0%	0.23	0.007	2701	2701	0.0%	0.06
12-10-20-20	0.16	904	904	0.0%	0.76	0.005	904	904	0.0%	0.10
12-10-20-25	0.09	396	396	0.0%	0.25	0.005	396	396	0.0%	0.06
12-10-20-30	0.11	335	335	0.0%	0.65	0.005	335	335	0.0%	0.53
12-10-20-35	0.13	669	669	0.0%	1.79	0.006	669	669	0.0%	0.27
12-10-20-50	0.10	811	811	0.0%	0.25	0.007	811	811	0.0%	0.04
13-10-20-20	0.12	1259	1259	0.0%	0.27	0.006	1259	1259	0.0%	0.03
13-10-20-25	0.12	669	669	0.0%	0.27	0.007	669	669	0.0%	0.07
13-10-20-30	0.18	580	580	0.0%	0.29	0.006	580	580	0.0%	0.06
13-10-20-35	0.10	266	266	0.0%	0.80	0.005	266	266	0.0%	0.10
13-10-20-50	0.11	678	678	0.0%	0.42	0.007	678	678	0.0%	0.08
14-10-20-20	0.16	633	633	0.0%	0.56	0.007	633	633	0.0%	0.08
14-10-20-25	0.11	1292	1292	0.0%	0.35	0.005	1292	1292	0.0%	0.03
14-10-20-30	0.13	574	574	0.0%	0.30	0.005	574	574	0.0%	0.04
14-10-20-35	0.13	2057	2057	0.0%	0.29	0.006	2057	2057	0.0%	0.03
14-10-20-50	0.13	355	355	0.0%	0.31	0.007	355	355	0.0%	0.07

Table C.8: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 15 to 19.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
15-10-20-20	0.14	560	560	0.0%	0.37	0.005	560	560	0.0%	0.04
15-10-20-25	0.32	1202	1202	0.0%	0.57	0.006	1202	1202	0.0%	0.07
15-10-20-30	0.11	1451	1451	0.0%	0.32	0.005	1451	1451	0.0%	0.02
15-10-20-35	0.15	1679	1679	0.0%	0.30	0.006	1679	1679	0.0%	0.03
15-10-20-50	0.15	623	623	0.0%	0.41	0.007	623	623	0.0%	0.03
16-10-20-20	0.32	736	736	0.0%	0.52	0.005	736	736	0.0%	0.08
16-10-20-25	0.16	1479	1479	0.0%	0.38	0.005	1479	1479	0.0%	0.03
16-10-20-30	0.16	1788	1788	0.0%	0.37	0.005	1788	1788	0.0%	0.06
16-10-20-35	0.14	2037	2037	0.0%	0.42	0.004	2037	2037	0.0%	0.05
16-10-20-50	0.16	1562	1562	0.0%	0.40	0.006	1562	1562	0.0%	0.04
17-10-20-20	0.24	632	632	0.0%	0.47	0.005	632	632	0.0%	0.03
17-10-20-25	0.18	1611	1611	0.0%	0.87	0.004	1611	1611	0.0%	0.02
17-10-20-30	0.18	424	424	0.0%	0.48	0.007	424	424	0.0%	0.03
17-10-20-35	0.24	2346.99	2346.99	0.0%	0.43	0.006	2347	2347	0.0%	0.05
17-10-20-50	0.17	2202	2202	0.0%	0.63	0.006	2202	2202	0.0%	0.09
18-10-20-20	0.31	1143	1143	0.0%	0.91	0.006	1143	1143	0.0%	0.09
18-10-20-25	0.23	291	291	0.0%	0.59	0.004	291	291	0.0%	0.02
18-10-20-30	0.25	1835	1835	0.0%	0.81	0.005	1835	1835	0.0%	0.05
18-10-20-35	0.18	991	991	0.0%	0.45	0.005	991	991	0.0%	0.04
18-10-20-50	0.22	550	550	0.0%	0.51	0.007	550	550	0.0%	0.04
19-10-20-20	0.25	905	905	0.0%	0.50	0.005	905	905	0.0%	0.03
19-10-20-25	0.28	535	535	0.0%	0.62	0.005	535	535	0.0%	0.03
19-10-20-30	0.18	1402	1402	0.0%	0.83	0.005	1402	1402	0.0%	0.04
19-10-20-35	0.22	691	691	0.0%	0.49	0.006	691	691	0.0%	0.06
19-10-20-50	0.23	2591	2591	0.0%	0.86	0.006	2591	2591	0.0%	0.06

Table C.9: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 20 to 29.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
20-20-100-200	1.48	14471	14471	0.0%	5.36	0.09	14471	14471	0.0%	1.15
20-20-130-200	1.51	31327	31327	0.0%	7.51	0.11	31327	31327	0.0%	4.26
20-20-150-200	1.70	34618	34618	0.0%	2.63	0.12	34618	34618	0.0%	1.10
21-20-100-200	1.75	2585	2585	0.0%	3.09	0.08	2585	2585	0.0%	0.98
21-20-130-200	2.05	4056	4056	0.0%	3.51	0.13	4056	4056	0.0%	1.02
21-20-150-200	1.88	27322	27322	0.0%	3.56	0.12	27322	27322	0.0%	1.06
22-20-100-200	1.96	4902	4902	0.0%	5.71	0.10	4902	4902	0.0%	1.36
22-20-130-200	1.99	12301	12301	0.0%	4.04	0.11	12301	12301	0.0%	0.94
22-20-150-200	2.12	10971	10971	0.0%	5.47	0.12	10971	10971	0.0%	1.40
23-20-100-200	1.90	14132	14132	0.0%	2.99	0.09	14132	14132	0.0%	0.75
23-20-130-200	1.94	21745	21745	0.0%	3.02	0.09	21745	21745	0.0%	1.17
23-20-150-200	2.20	23358	23358	0.0%	5.03	0.12	23358	23358	0.0%	1.93
24-20-100-200	2.37	10155	10155	0.0%	6.28	0.08	10155	10155	0.0%	1.32
24-20-130-200	3.51	3486	3486	0.0%	13.02	0.14	3486	3486	0.0%	5.79
24-20-150-200	3.26	18156	18156	0.0%	9.43	0.15	18156	18156	0.0%	2.36
25-20-100-200	2.33	9335	9335	0.0%	5.93	0.07	9335	9335	0.0%	1.09
25-20-130-200	2.64	33905	33905	0.0%	4.07	0.12	33905	33905	0.0%	0.99
25-20-150-200	2.91	32508	32508	0.0%	6.05	0.14	32508	32508	0.0%	1.80
26-20-100-200	2.64	9815	9815	0.0%	4.16	0.08	9815	9815	0.0%	0.73
26-20-130-200	2.76	16892	16892	0.0%	3.93	0.10	16892	16892	0.0%	0.70
26-20-150-200	3.94	3731	3731	0.0%	21.92	0.14	3731	3731	0.0%	5.35
27-20-100-200	3.31	21090	21090	0.0%	4.54	0.08	21090	21090	0.0%	0.76
27-20-130-200	3.95	8456	8456	0.0%	29.05	0.14	8456	8456	0.0%	6.51
27-20-150-200	3.29	26704	26704	0.0%	6.69	0.12	26704	26704	0.0%	1.94
28-20-100-200	3.23	3641	3641	0.0%	42.24	0.08	3641	3641	0.0%	8.75
28-20-130-200	3.32	25355	25355	0.0%	4.63	0.10	25355	25355	0.0%	1.19
28-20-150-200	3.73	11222	11222	0.0%	5.49	0.12	11222	11222	0.0%	1.15
29-20-100-200	3.69	5994	5994	0.0%	7.48	0.08	5994	5994	0.0%	1.56
29-20-130-200	3.60	8978	8978	0.0%	72.53	0.12	8978	8978	0.0%	36.02
29-20-150-200	3.91	7977	7977	0.0%	6.73	0.11	7977	7977	0.0%	0.76

Table C.10: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 30 to 39.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
30-30-200-300	8.23	14512	14512	0.0%	16.41	0.36	14512	14512	0.0%	3.96
30-30-230-300	9.41	47054	47054	0.0%	21.33	0.46	47054	47054	0.0%	6.41
30-30-250-300	10.03	68998	68998	0.0%	108.11	0.51	68998	68998	0.0%	71.58
31-30-200-300	8.46	51221	51221	0.0%	11.79	0.30	51221	51221	0.0%	2.08
31-30-230-300	9.68	79441	79441	0.0%	1539.31	0.41	79441	79441	0.0%	382.25
31-30-250-300	10.34	79063	79063	0.0%	13.13	0.49	79063	79063	0.0%	4.01
32-30-200-300	9.11	46435	46435	0.0%	39.08	0.35	46435	46435	0.0%	9.51
32-30-230-300	10.83	23497	23497	0.0%	18.04	0.40	23497	23497	0.0%	2.51
32-30-250-300	17.02	6279	6279	0.0%	394.39	0.57	6279	6279	0.0%	176.37
33-30-200-300	9.86	55911	55911	0.0%	254.35	0.35	55911	55911	0.0%	21.98
33-30-230-300	11.13	48194	48194	0.0%	32.99	0.40	48194	48194	0.0%	6.21
33-30-250-300	11.59	33569	33569	0.0%	65.80	0.41	33569	33569	0.0%	16.85
34-30-200-300	14.44	4623	4623	0.0%	63.03	0.45	4623	4623	0.0%	14.86
34-30-230-300	12.19	62724	62724	0.0%	56.14	0.45	62724	62724	0.0%	11.04
34-30-250-300	12.77	16985	16985	0.0%	21.54	0.44	16985	16985	0.0%	4.00
35-30-200-300	11.06	29110	29110	0.0%	170.60	0.34	29110	29110	0.0%	34.44
35-30-230-300	13.41	76015	76015	0.0%	45.84	0.40	76015	76015	0.0%	6.39
35-30-250-300	13.90	47613	47613	0.0%	61.65	0.44	47613	47613	0.0%	13.50
36-30-200-300	13.64	40288	40288	0.0%	29.92	0.39	40288	40288	0.0%	4.82
36-30-230-300	14.57	52902	52902	0.0%	28.51	0.44	52902	52902	0.0%	6.79
36-30-250-300	14.50	62932	62932	0.0%	21.88	0.47	62932	62932	0.0%	2.76
37-30-200-300	13.07	57422	57422	0.0%	16.96	0.29	57422	57422	0.0%	6.33
37-30-230-300	16.65	7769	7769	0.0%	1250.38	0.39	7769	7769	0.0%	273.13
37-30-250-300	16.07	41580	41580	0.0%	48.15	0.41	41580	41580	0.0%	7.76
38-30-200-300	14.08	28805	28805	0.0%	465.01	0.34	28805	28805	0.0%	75.33
38-30-230-300	15.72	73698	73698	0.0%	25.18	0.44	73698	73698	0.0%	3.72
38-30-250-300	18.31	24709	24709	0.0%	26.90	0.48	24709	24709	0.0%	4.05
39-30-200-300	18.71	4989	4989	0.0%	69.13	0.39	4989	4989	0.0%	9.22
39-30-230-300	17.87	67958	67958	0.0%	238.48	0.43	67958	67958	0.0%	44.26
39-30-250-300	17.57	63894	63894	0.0%	60.95	0.46	63894	63894	0.0%	9.59

Table C.11: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 40 to 49.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
40-36-130-500	15.23	48485	48485	0.0%	47.90	0.46	48485	48485	0.0%	11.66
40-36-150-500	17.29	28510	28510	0.0%	32.65	0.62	28510	28510	0.0%	8.89
40-36-170-500	18.22	76106	76106	0.0%	52.27	0.69	76106	76106	0.0%	11.44
41-36-130-500	18.11	4349	4349	0.0%	113.69	0.48	4349	4349	0.0%	27.03
41-36-150-500	19.53	63691	63691	0.0%	24.16	0.62	63691	63691	0.0%	5.85
41-36-170-500	20.81	54550	54550	0.0%	25.21	0.70	54550	54550	0.0%	4.64
42-36-130-500	19.18	15598	15598	0.0%	42.48	0.47	15598	15598	0.0%	6.05
42-36-150-500	22.40	8178	8178	0.0%	26.89	0.78	8178	8178	0.0%	7.63
42-36-170-500	21.62	36376	36376	0.0%	80.60	0.67	36376	36376	0.0%	62.20
43-36-130-500	20.24	35988	35988	0.0%	25.25	0.46	35988	35988	0.0%	4.94
43-36-150-500	21.36	53269	53269	0.0%	138.62	0.61	53269	53269	0.0%	30.75
43-36-170-500	21.77	37484	37484	0.0%	65.34	0.68	37484	37484	0.0%	15.47
44-36-130-500	20.28	55748	55748	0.0%	79.92	0.47	55748	55748	0.0%	16.84
44-36-150-500	22.88	70064	70064	0.0%	39.81	0.61	70064	70064	0.0%	7.70
44-36-170-500	25.29	37247	37247	0.0%	128.38	0.68	37247	37247	0.0%	26.54
45-36-130-500	21.41	35017	35017	0.0%	38.40	0.46	35017	35017	0.0%	7.64
45-36-150-500	22.82	35095	35095	0.0%	69.02	0.53	35095	35095	0.0%	13.67
45-36-170-500	25.23	67070	67070	0.0%	35.68	0.60	67070	67070	0.0%	6.51
46-36-130-500	26.07	4479	4479	0.0%	393.58	0.61	4479	4479	0.0%	69.61
46-36-150-500	24.76	25481	25481	0.0%	232.55	0.61	25481	25481	0.0%	62.85
46-36-170-500	28.27	45355	45355	0.0%	37.83	0.67	45355	45355	0.0%	6.55
47-36-130-500	34.85	4825	4825	0.0%	69.29	0.69	4825	4825	0.0%	11.31
47-36-150-500	26.57	54139	54139	0.0%	76.43	0.60	54139	54139	0.0%	42.31
47-36-170-500	30.42	46154	46154	0.0%	78.83	0.69	46154	46154	0.0%	13.71
48-36-130-500	28.28	26069	26069	0.0%	91.27	0.59	26069	26069	0.0%	15.70
48-36-150-500	33.14	4848	4848	0.0%	118.50	0.72	4848	4848	0.0%	20.66
48-36-170-500	29.44	68476	68476	0.0%	163.78	0.59	68476	68476	0.0%	26.56
49-36-130-500	28.81	38305	38305	0.0%	37.78	0.52	38305	38305	0.0%	5.06
49-36-150-500	40.63	4944	4944	0.0%	121.98	0.70	4944	4944	0.0%	19.01
49-36-170-500	34.51	4676	4676	0.0%	72.37	0.69	4676	4676	0.0%	14.07

Table C.12: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 50 to 54.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
50-36-100-700	23.56	50961	50961	0.0%	34.71	0.69	50961	50961	0.0%	7.61
50-36-130-700	29.24	57051.5	57048	**0.0001%**	**7200.00**	1.02	570489	57048	0.0%	5822.36
50-36-150-700	31.62	22656	22656	0.0%	68.35	1.19	22656	22656	0.0%	20.72
50-36-180-700	34.10	62122	62122	0.0%	627.10	1.39	62122	62122	0.0%	245.91
50-36-200-700	34.85	28843	28843	0.0%	37.93	1.50	28843	28843	0.0%	10.51
50-36-250-700	40.64	111448	111444	**0.0001%**	**7200.00**	1.91	111444	111444	0.0%	761.11
51-36-100-700	24.03	17676	17676	0.0%	99.31	0.59	17676	17676	0.0%	25.22
51-36-130-700	30.62	62845	62845	0.0%	42.11	1.02	62845	62845	0.0%	9.77
51-36-150-700	30.51	61608	61608	0.0%	171.96	1.16	61608	61608	0.0%	56.61
51-36-180-700	37.18	45116	45116	0.0%	1798.33	1.39	45116	45116	0.0%	777.42
51-36-200-700	38.92	69124	69124	0.0%	55.16	1.72	69124	69124	0.0%	10.71
51-36-250-700	49.22	29392	29392	0.0%	772.57	2.18	29392	29392	0.0%	268.88
52-36-100-700	25.76	47273	47273	0.0%	38.97	0.68	47273	47273	0.0%	7.48
52-36-130-700	33.24	60081	60081	0.0%	63.88	1.02	60081	60081	0.0%	11.91
52-36-150-700	28.20	75988	75988	0.0%	46.27	0.89	75988	75988	0.0%	7.81
52-36-180-700	37.01	54693	54693	0.0%	208.32	1.22	54693	54693	0.0%	73.72
52-36-200-700	43.32	33908	33908	0.0%	91.11	1.53	33908	33908	0.0%	18.60
52-36-250-700	52.17	8242	8242	0.0%	224.81	2.19	8242	8242	0.0%	9.48
53-36-100-700	26.39	18727	18727	0.0%	150.27	0.58	18727	18727	0.0%	58.73
53-36-130-700	31.03	36388	36388	0.0%	51.06	0.88	36388	36388	0.0%	12.69
53-36-150-700	38.67	63060	63060	0.0%	55.33	1.15	63060	63060	0.0%	11.14
53-36-180-700	38.71	28785	28785	0.0%	1023.86	1.39	28785	28785	0.0%	314.93
53-36-220-700	42.52	101729	101729	0.0%	105.90	1.44	101729	101729	0.0%	43.48
53-36-250-700	49.47	104922	104922	0.0%	65.19	1.90	104922	104922	0.0%	18.85
54-36-100-700	29.48	23551	23551	0.0%	137.10	0.68	23551	23551	0.0%	31.67
54-36-130-700	33.05	39535	39535	0.0%	45.46	1.01	39535	39535	0.0%	9.91
54-36-150-700	38.63	4722	4722	0.0%	81.05	1.16	4722	4722	0.0%	24.17
54-36-180-700	40.40	36084	36084	0.0%	107.63	1.56	36084	36084	0.0%	35.34
54-36-200-700	41.92	54198	54198	0.0%	45.06	1.31	54198	54198	0.0%	14.89
54-36-250-700	57.90	8531	8531	0.0%	1418.64	2.14	8531	8531	0.0%	469.17

Table C.13: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances with terminals ranging from 55 to 59.

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
55-36-100-700	29.75	28195	28195	0.0%	100.46	0.68	28195	28195	0.0%	16.99
55-36-130-700	47.77	4506	4506	0.0%	73.97	1.02	4506	4506	0.0%	12.71
55-36-150-700	40.81	7853	7853	0.0%	49.44	1.16	7853	7853	0.0%	14.83
55-36-180-700	43.37	40857	40857	0.0%	109.13	1.37	40857	40857	0.0%	27.87
55-36-200-700	45.23	29966	29966	0.0%	120.55	1.52	29966	29966	0.0%	30.83
55-36-250-700	55.72	54658	54658	0.0%	108.08	2.15	54658	54658	0.0%	34.86
56-36-100-700	35.63	3481	3481	0.0%	54.07	0.69	3481	3481	0.0%	5.23
56-36-130-700	52.44	4433	4433	0.0%	57.63	1.03	4433	4433	0.0%	13.00
56-36-150-700	41.97	11529	11529	0.0%	98.02	1.17	11529	11529	0.0%	20.91
56-36-180-700	48.28	44014	44014	0.0%	244.86	1.37	44014	44014	0.0%	73.35
56-36-200-700	51.13	71438	71438	0.0%	65.10	1.32	71438	71438	0.0%	14.93
56-36-250-700	54.70	85911	85911	0.0%	1942.71	2.12	85911	85911	0.0%	546.20
57-36-100-700	33.38	12646	12646	0.0%	49.92	0.68	12646	12646	0.0%	7.60
57-36-130-700	37.13	59518	59518	0.0%	153.09	0.87	59518	59518	0.0%	30.21
57-36-150-700	40.73	51978	51978	0.0%	80.27	1.00	51978	51978	0.0%	13.53
57-36-180-700	47.38	62345	62345	0.0%	62.52	1.54	62345	62345	0.0%	13.76
57-36-200-700	52.76	91807	91807	0.0%	57.93	1.72	91807	91807	0.0%	10.89
57-36-250-700	62.00	47464	47464	0.0%	63.98	2.12	47464	47464	0.0%	19.03
58-36-100-700	48.58	31050	31050	0.0%	99.66	1.19	31050	31050	0.0%	27.76
58-36-130-700	31.85	18790	18790	0.0%	49.34	0.58	18790	18790	0.0%	9.71
58-36-150-700	53.58	64722	64722	0.0%	51.38	1.50	64722	64722	0.0%	11.63
58-36-180-700	42.30	49225	49225	0.0%	164.57	0.98	49225	49225	0.0%	34.26
58-36-200-700	62.03	24632	24632	0.0%	80.03	2.12	24632	24632	0.0%	30.48
58-36-250-700	46.64	116424	116424	0.0%	2271.38	1.14	116424	116424	0.0%	632.19
59-36-100-700	46.65	3566	3566	0.0%	511.11	0.78	3566	3566	0.0%	99.67
59-36-130-700	50.34	4956	4956	0.0%	1147.60	1.28	4956	4956	0.0%	201.27
59-36-150-700	60.56	5374	5374	0.0%	79.93	1.12	5374	5374	0.0%	14.84
59-36-180-700	52.15	83504	83504	0.0%	256.82	1.36	83504	83504	0.0%	56.59
59-36-200-700	55.35	46808	46808	0.0%	73.87	1.31	46808	46808	0.0%	14.88
59-36-250-700	63.43	96410	96410	0.0%	82.01	1.86	96410	96410	0.0%	24.60

Table C.14: Results from solving the arc-based and node-based formulations via Branch-and-Price in instances from Vasco and Morabito (2016b).

Instance	*Arc-demand*					*Node-demand*				
	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}	T_{Root}	UB	ILP_{OV}	Gap	T_{BP}
53-36-130-300-p1	32.98	17437.6	17437.6	0.0%	46.89	0.53	17437.6	17437.6	0.0%	0.45
53-36-130-300-p2	29.10	19260	19260	0.0%	29.10	0.24	19260	19260	0.0%	0.24
53-36-130-300-p3	31.03	16633.8	16633.8	0.0%	409.09	0.41	16634.2	16633.8	0.0%	1.24
53-36-130-300-p4	36.83	19560	19560	0.0%	36.83	0.34	19560	19560	0.0%	0.34
53-36-130-300-p5	29.08	18169.2	18169.2	0.0%	29.08	0.20	18169.2	18169.2	0.0%	0.20
53-36-130-300-p6	33.00	19969.4	19969.4	0.0%	33.00	0.28	19969.4	19969.4	0.0%	0.28
53-36-130-300-p7	31.51	19213.8	19213.8	0.0%	31.51	0.30	19213.8	19213.8	0.0%	0.30
53-36-130-300-p8	29.23	18472.6	18472.6	0.0%	127.94	0.17	18472.6	18472.6	0.0%	1.46
53-36-130-300-p9	36.74	15371.4	15371.4	0.0%	36.74	0.30	15371.4	15371.4	0.0%	0.30
53-36-130-300-p10	25.20	18344.8	18344.8	0.0%	25.20	0.19	18344.8	18344.8	0.0%	0.19
53-36-130-300-p11	36.01	16799.6	16799.6	0.0%	46.50	0.17	16799.7	16799.6	0.0%	2.04
53-36-130-300-p12	34.73	22008.4	22008.4	0.0%	34.73	0.33	22008.4	22008.4	0.0%	0.33
53-36-130-300-p13	37.51	19628.2	19628.2	0.0%	80.58	0.15	19628.2	19628.2	0.0%	0.93
53-36-130-300-p14	31.77	19616.6	19616.6	0.0%	31.77	0.22	19616.6	19616.6	0.0%	0.22
53-36-130-300-p15	38.57	20673.2	20673.2	0.0%	38.57	0.25	20673.2	20673.2	0.0%	0.25
53-36-130-300-p16	36.01	17796.2	17796.2	0.0%	36.01	0.23	17796.2	17796.2	0.0%	0.23
53-36-130-300-p17	33.52	17345.2	17345.2	0.0%	33.52	0.18	17345.2	17345.2	0.0%	0.18
53-36-130-300-p18	36.62	17849.2	17849.2	0.0%	82.99	0.51	17849.2	17849.2	0.0%	0.78
53-36-130-300-p19	37.23	18190.6	18190.6	0.0%	37.23	0.21	18190.6	18190.6	0.0%	0.21
53-36-130-300-p20	36.26	20754.4	20754.4	0.0%	36.26	0.24	20754.4	20754.4	0.0%	0.24
53-36-130-300-p21	44.80	16953.2	16953.2	0.0%	44.80	0.33	16953.2	16953.2	0.0%	0.33
53-36-130-300-p22	37.64	18699.2	18699.2	0.0%	37.64	0.36	18699.2	18699.2	0.0%	0.36
53-36-130-300-p23	34.35	21525.6	21525.6	0.0%	34.35	0.28	21525.6	21525.6	0.0%	0.28
53-36-130-300-p24	27.34	18266.2	18266.2	0.0%	27.34	0.17	18266.2	18266.2	0.0%	0.17
53-36-130-300-p25	33.85	17064.8	17064.8	0.0%	33.85	0.28	17064.8	17064.8	0.0%	0.28
53-36-130-300-p26	30.16	20324.4	20324.4	0.0%	30.16	0.23	20324.4	20324.4	0.0%	0.23
53-36-130-300-p27	27.10	20003	20003	0.0%	27.10	0.21	20003	20003	0.0%	0.21
53-36-130-300-p28	37.92	17956	17956	0.0%	37.92	0.24	17956	17956	0.0%	0.24
53-36-130-300-p29	30.23	19074.6	19074.6	0.0%	30.23	0.22	19074.6	19074.6	0.0%	0.22
53-36-130-300-p30	33.68	16464.6	16464.6	0.0%	33.68	0.22	16464.6	16464.6	0.0%	0.22

Bibliography

Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42 – 54, 2005. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2004.04.002.

Ravindra Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, N.J, 1993. ISBN 978-0136175490.

Aldair Alvarez and Pedro Munari. An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, 83:1–12, 2017. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2017.02.001.

Filipe Alvelos and J. M. Valerio de Carvalho. An extended model and a column generation algorithm for the planar multicommodity flow problem. *Networks*, 50(1):3–16, 2007. doi: 10.1002/net.20161.

Jay E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20(1):1–66, Dec 1989. doi: 10.1007/BF02216922.

Sundararajan Arunapuram, Kamlesh Mathur, and Daniel Solow. Vehicle routing and scheduling with full truckloads. *Transportation Science*, 37(2):170–182, 2003. doi: 10.1287/trsc.37.2.170.15248.

Ruibin Bai, Ning Xue, Jianjun Chen, and Gethin Wyn Roberts. A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. *Transportation Research Part B: Methodological*, 79: 134 – 148, 2015. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2015.06.002.

Cynthia Barnhart, Christopher A. Hane, Ellis L. Johnson, and Gabriele Sigismondi. A column generation and partitioning approach for multi-commodity flow problems. *Telecommunication Systems*, 3:239–258, 1994. doi: 10.1007/BF02110307.

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000. doi: 10.1287/opre.48.2.318.12378.

Dimitri P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, Massachusetts, USA, 1998. ISBN 1-886529-02-7.

Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

Richard Chase, Nicholas J. Aquilano, and F. Robert Jacobs. *Production and operations management : manufacturing and services*. Irwin/McGraw-Hill, Boston, Mass, 1998. ISBN 007561278X.

Raymond K. Cheung and Warren B. Powell. An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management. *Operations Research*, 44(6): 951–963, 1996. doi: 10.1287/opre.44.6.951.

Luca Coslovich, Raffaele Pesenti, and Walter Ukovich. Minimizing fleet operating costs for a container transportation company. *European Journal of Operational Research*, 171(3):776 – 786, 2006. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2004.09.005.

Alysson M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32(6):1429 – 1450, 2005. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2003.11.012.

Alysson M. Costa, Jean François Cordeau, and Bernard Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 242 (3):371–392, 2009. doi: 10.1007/s10589-007-9122-0.

Teodor Gabriel Crainic. *Long-Haul Freight Transportation*, pages 451–516. Springer US, Boston, MA, 2003.

Cesar Cruz, Pedro Munari, and Reinaldo Morabito. Bounds for the vehicle allocation problem. *DYNA*, 86(208):329–335, 2019. ISSN 2346-2183. doi: 10.15446/dyna.v86n208.68504.

Cesar Alvarez Cruz, Pedro Munari, and Reinaldo Morabito. A branch-and-price method for the vehicle allocation problem. *Computers & Industrial Engineering*, 149:106745, 2020. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2020.106745.

Cesar Dario Alvarez Cruz. Abordagens de otimização para o problema de alocação dinâmica de veículos no contexto de transporte rodoviário de carga no brasil. Master's thesis, Universidade Federal de São Carlos, S.P., Brazil, 2017.

P. J. Dejax and T. G. Crainic. Survey paper–a review of empty flows and fleet management models in freight transportation. *Transportation Science*, 21(4):227–248, 1987. ISSN 0041-1655.

Martin Desrochers and François Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR: Information Systems and Operational Research*, 26(3):191–212, 1988. doi: 10.1080/03155986.1988.11732063.

Jacques Desrosiers, François Soumis, and Martin Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984. doi: 10.1002/net.3230140406.

Jacques Desrosiers, Gilbert Laporte, Michel Sauve, François Soumis, and Serge Taillefer. Vehicle routing with full loads. *Computers & Operations Research*, 15(3):219 – 226, 1988. ISSN 0305-0548. doi: https://doi.org/10.1016/0305-0548(88)90034-2.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan 2002. ISSN 1436-4646. doi: 10.1007/s101070100263.

Alan L. Erera, Juan C. Morales, and Martin Savelsbergh. Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483, 2009.

Linos F. Frantzeskakis and Warren B. Powell. A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems. *Transportation Science*, 24(1):40–57, 1990. ISSN 0041-1655.

Michel Gendreau, Jenny Nossack, and Erwin Pesch. Mathematical formulations for a 1-full-truckload pickup-and-delivery problem. *European Journal of Operational Research*, 242(3):1008 – 1016, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2014.10.053.

Bernard Gendron and Mathieu Larose. Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization*, 2(1):55–75, 2014. doi: 10.1007/s13675-014-0020-9.

Giampaolo Ghiani, Gilbert Laporte, and Roberto Musmanno. *Introduction to logistics systems planning and control*. J. Wiley, Hoboken, NJ, USA, 2004. ISBN 978-0470849163.

Gregory A. Godfrey and Warren B. Powell. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science*, 47(8):1101–1112, 2001. doi: 10.1287/mnsc.47.8.1101.10231.

Gregory A. Godfrey and Warren B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science*, 36(1):21–39, 2002a. doi: 10.1287/trsc.36.1.21.570.

Gregory A. Godfrey and Warren B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, ii: Multiperiod travel times. *Transportation Science*, 36(1):40–54, 2002b. doi: 10.1287/trsc.36.1.40.572.

Jacek Gondzio and Pedro Munari. Column generation and branch-and-price with interior point methods. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 3(1):41 – 51, 2015. ISSN 0377-2217. doi: http://dx.doi.org/10.5540/03.2015.003.01.0525.

Jacek Gondzio and Robert. Sarkissian. Column generation with a primal-dual method. Technical report, Logilab, HEC Geneva, University of Geneva, 1996.

Jacek Gondzio, Pablo González-Brevis, and Pedro Munari. New developments in the primal-dual column generation technique. *European Journal of Operational Research*, 224(1):41 – 51, 2013. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2012.07.024.

Jacek Gondzio, Pablo González-Brevis, and Pedro Munari. Large-scale optimization with the primal-dual column generation method. *Mathematical Programming Computation*, 8(1):47–82, 2016. ISSN 1867-2957. doi: 10.1007/s12532-015-0090-6.

Manfred Gronalt, Richard F. Hartl, and Marc Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, 151(3):520 – 535, 2003. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(02)00650-1.

Kaj Holmberg and Di Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1):42–57, 2003. doi: 10.1287/ijoc.15.1.42.15151.

Richard E Hughes, Warren B. Powell, and Ann Arbor. Mitigating end effects in the dynamic vehicle allocation model. *Management Science*, 34(7):859–879, 1988. doi: 10.1287/mnsc.34.7.859.

Kim L. Jones, Irvin J. Lustig, Judith M. Farvolden, and Warren B. Powell. Multicommodity network flows: The impact of formulation on decomposition. *Mathematical Programming*, 62(1-3):95–117, 1993. ISSN 00255610.

Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015. doi: 10.1080/10556788.2014.895828.

Chungmok Lee, Kyungsik Lee, and Sungsoo Park. Benders decomposition approach for the robust network design problem with flow bifurcations. *Networks*, 62(1):1–16, 2013. doi: 10.1002/net.21486.

Jian Li and Wenhua Lu. Full truckload vehicle routing problem with profits. *Journal of Traffic and Transportation Engineering (English Edition)*, 1(2):146 – 152, 2014. ISSN 2095-7564. doi: https://doi.org/10.1016/S2095-7564(15)30099-4.

Ran Liu, Zhibin Jiang, Richard Y.K. Fung, Feng Chen, and Xiao Liu. Two-phase heuristic algorithms for full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration. *Computers & Operations Research*, 37(5):950 – 959, 2010a. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2009.08.002. Disruption Management.

Ran Liu, Zhibin Jiang, Xiao Liu, and Feng Chen. Task selection and routing problems in collaborative truckload transportation. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):1071 – 1085, 2010b. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2010.05.003.

Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, nov 2005. ISSN 0030-364X. doi: 10.1287/opre.1050.0234.

Gino Marchet, Sara Perotti, and Riccardo Mangiaracina. Modelling the impacts of ict adoption for intermodal transportation. *International Journal of Physical Distribution & Logistics Management*, 42 (2):110–127, 2012. doi: 10.1108/09600031211219645.

Luigi Moccia, Jean François Cordeau, Maria Flavia Monaco, and Marcello Sammarra. A column generation heuristic for a dynamic generalized assignment problem. *Computers & Operations Research*, 36 (9):2670 – 2681, 2009. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2008.11.022.

Pedro Munari and Jacek Gondzio. Using the primal-dual interior point algorithm within the branch-price-and-cut method. *Computers & Operations Research*, 40(8):2026–2036, 2013.

Pedro Munari, Alfredo Moreno, Jonathan De La Vega, Douglas Alem, Jacek Gondzio, and Reinaldo Morabito. The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, 2019. doi: 10.1287/trsc.2018.0886.

Marc Peeters and Leo Kroon. Circulation of railway rolling stock: a branch-and-price approach. *Computers & Operations Research*, 35(2):538 – 556, 2008. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2006.03.019. Part Special Issue: Location Modeling Dedicated to the memory of Charles S. ReVelle.

Victor Pillac, Michel Gendreau, Christelle GuÃ©ret, and AndrÃ©s L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1 – 11, 2013. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2012.08.015.

Warren B. Powell. A stochastic model of the dynamic vehicle allocation problem. *Transportation Science*, 20(2):117–129, 1986. ISSN 0041-1655.

Warren B. Powell and Tassio A. Carvalho. Dynamic control of logistics queueing networks for large-scale fleet management. *Transportation Science*, 32(2):90–109, 1998.

Warren B. Powell and Raymond K.-M. Cheung. A network recourse decomposition method for dynamic networks with random arc capacities. *Networks*, 24(7):369–384, 1994. ISSN 1097-0037. doi: 10.1002/net.3230240703.

Warren B. Powell, Yosef Sheffi, and Sebastien Thiriez. The dynamic vehicle allocation problem with uncertain demands. In J. Volmuller and R. Hamerslag, editors, *Proceedings of the Ninth International Symposium on Transportation and Traffic Theory*, pages 357–374, 1984.

Warren B. Powell, Patrick Jaillet, and Amadeo. Odoni. Stochastic and dynamic networks and routing. *Network Routing*, 8:141–295, 1995.

Warren B. Powell, Wayne Snow, and Raymond K. Cheung. Adaptive labeling algorithms for the dynamic assignment problem. *Transportation Science*, 34(1):50–66, 2000. doi: 10.1287/trsc.34.1.50.12280.

Warren B. Powell, Joel A. Shapiro, and Hugo P. Simão. An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science*, 36(2):231–249, 2002. doi: 10.1287/trsc.36.2.231.561.

Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.12.005.

Priscilla Cristina Cabral Ribeiro and Karine Araújo Ferreira. Logística e transportes : Uma discussão sobre os modais de transporte e o panorama brasileiro . *XXII Encontro Nacional de Engenharia de Produção - ENEGEP*, pages 1–8, 2002.

Ralph Tyrrel Rockafellar. *Network flows and monotropic optimization.* Athena Scientific, Belmont, Mass, 1998. ISBN 1-886529-06-X.

J. Roy. Recent trends in logistics and the need for real-time decision tools in the trucking industry. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, Jan 2001.

Alessandro Serrano Colavite and Fabio Konishi. A matriz do transporte no brasil: uma análise comparativa para a competitividade. In *Simpósio de Excelência em Gestão e Tecnologia*, Oct 2015.

Suresh Sethi and Gerhard Sorger. A theory of rolling horizon decision making. *Annals of Operations Research*, 29(1):387–415, 1991. ISSN 1572-9338. doi: 10.1007/BF02283607.

Ning Shi, Haiqing Song, and Warren B. Powell. The dynamic fleet management problem with uncertain demand and customer chosen service level. *International Journal of Production Economics*, 148 (Supplement C):110 – 121, 2014. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2013.09.010.

Michael Z. Spivey and Warren B. Powell. The dynamic assignment problem. *Transportation Science*, 38 (4):399–419, 2004. doi: 10.1287/trsc.1030.0073.

Varadharajan Sridhar and June S. Park. Benders-and-cut algorithm for fixed-charge capacitated network design problem. *European Journal of Operational Research*, 125(3):622 – 632, 2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(99)00272-6.

François Vanderbeck and Laurence A. Wolsey. An exact algorithm for ip column generation. *Operations Research Letters*, 19(4):151 – 159, 1996. ISSN 0167-6377. doi: https://doi.org/10.1016/0167-6377(96)00033-8.

François Vanderbeck. *Implementing Mixed Integer Column Generation*, pages 331–358. Springer US, Boston, MA, 2005. ISBN 978-0-387-25486-9. doi: 10.1007/0-387-25486-2_12.

Rejane Arinos Vasco. *Otimização na alocação dinâmica de veículos no transporte rodoviário de cargas completas entre terminais*. PhD thesis, Universidade Federal de São Carlos, 2012.

Rejane Arinos Vasco and Reinaldo Morabito. Otimização na alocação dinâmica de veículos no transporte rodoviário de cargas completas entre terminais. *Gestão e Produção*, 21:271 – 284, 06 2014. ISSN 0104-530X.

Rejane Arinos Vasco and Reinaldo Morabito. Dimensionamento e alocação dinâmica de veículos no transporte rodoviário de cargas completas entre terminais. *Production*, 26:430 – 444, 06 2016a. ISSN 0103-6513.

Rejane Arinos Vasco and Reinaldo Morabito. The dynamic vehicle allocation problem with application in trucking companies in brazil. *Computers & Operations Research*, 76:118 – 133, 2016b. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2016.04.022.