MODELOS E ALGORITMOS PARA PROBLEMAS INTEGRADOS DE DISTRIBUIÇÃO E ROTEAMENTO

FERNANDO AFONSO SANTOS

MODELOS E ALGORITMOS PARA PROBLEMAS INTEGRADOS DE DISTRIBUIÇÃO E ROTEAMENTO

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Geraldo Robson Mateus

Coorientador: Alexandre Salles da Cunha

Belo Horizonte

Dezembro de 2012

FERNANDO AFONSO SANTOS

MODELS AND ALGORITHMS FOR THE INTEGRATED ROUTING AND DISTRIBUTION PROBLEMS

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Geraldo Robson Mateus

Co-advisor: Alexandre Salles da Cunha

Belo Horizonte

December 2012

Santos, Fernando Afonso.
S237m Modelos e algoritmos para problemas integrados de distribuição e roteamento / Fernando Afonso Santos — Belo Horizonte, 2012.
xx, 116 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação.
Orientador: Geraldo Robson Mateus
Coorientador: Alexandre Salles da Cunha

1. Computação - Teses. 2. Otimização combinatória. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*61(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Modelos e algoritmos para problemas integrados de distribuição e roteamento
(Models and algorithms for the integrated routing and distribution problems)

FERNANDO AFONSO SANTOS

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. GERALDO ROBSON MATEUS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ALEXANDRE SALLES DA CUNHA - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. CELSO DA CRUZ CARNEIRO RIBEIRO
Departamento de Computação - UFF

PROF. FLÁVIO KEIDI MIYAZAWA
Instituto de Computação - UNICAMP

PROF. HENRIQUE PACCA LOUREIRO LUNA
Instituto de Computação - UFAL

PROF. MARCUS VINÍCIUS SOLEDADE POGGI DE ARAGÃO
Departamento de Informática - PUC/Rio

Belo Horizonte, 14 de dezembro de 2012.

Dedico este trabalho à minha avó Maria Izidoria, que sempre me serviu de motivação, mesmo infelizmente não estando mais entre nós.

Abstract

Since the Vehicle Routing Problem (VRP) was introduced by Dantzig and Ramser, it became one of the most studied problems in Combinatorial Optimization. Different solution approaches were proposed over the past decades to solve the VRP and its variants. In this thesis, we discuss about two VRP variants, resulting from the integration of VRPs with distribution problems.

The first problem takes place by integrating the VRP with loading/unloading decisions in a Cross-Docking warehouse, which allows the consolidation of loads between their pickup and delivery. The problem of dealing with routing and distribution decisions at the Cross-Docking is named the Vehicle Routing Problem with Cross-Docking (VRPCD). We introduced two Integer Programming (IP) formulations for the VRPCD and respective branch-and-price (BP) algorithms to evaluate them. We also consider a slightly different approach for solving the VRPCD, where vehicles are allowed to bypass the Cross-Docking and loads eventually are not consolidated. We call this problem as the Pickup and Delivery Problem with Cross-Docking (PDPCD). We also introduced an IP model for the PDPCD and a BP algorithm to solve it.

The second problem we deal in this thesis arises in multi-echelon systems. The Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP) arises where loads must be shipped from a depot to customers passing through intermediate warehouses named satellites. Loads are shipped from the depot to satellites, where they are consolidated before to be delivered to their respective customers. We propose an IP formulation for 2E-CVRP which holds an exponential number of variables. In addition, we introduce four families of valid inequalities for the problem. To solve such a formulation, including valid inequalities, we implement a Branch-and-cut-and-price (BCP) algorithm. Our computational results show that BCP algorithm evaluates stronger lower and upper bounds than previous algorithms for 2E-CVRP and also provides new optimality certificates for different instances of the literature.

Palavras-chave: Vehicle Routing Problems, Distribution Problems, Column Generation, Branch-and-price, Cutting planes.

Resumo

Após ter sido introduzido por Dantzig and Ramser, o Problema de Roteamento de Veículos (PRV) se tornou um dos problemas mais estudados em Otimização Combinatória. Diferentes estratégias de solução foram propostas ao longo do tempo para solucionar o PRV e suas variações. Nesta tese, são discutidos dois problemas resultantes da integração do PRV com problemas de distribuição de mercadorias.

O primeiro problema a ser discutido surge da integração do PRV com decisões sobre carga/descarga de mercadorias em um *Cross-Docking*, que é um armazém que permite armazenamento de curta duração e a consolidação das cargas entre sua coleta e a entrega. O problema resultante desta integração é denominado Problema de Roteamento de Veículos com *Cross-Docking* (PRVCD). São apresentados dois modelos de programação inteira para o PRVCD e respectivos algoritmos *Branch-and-Price* (BP) para solucionar tais modelos. Será discutida também uma abordagem alternativa aos modelos tradicionais do PRVCD, na qual os veículos podem evitar passar pelo *Cross-Docking* na coleta/engrega de cargas, sem consolidá-las. Denomina-se esta nova abordagem de Problema de Coleta e Entrega com *Cross-Docking* (PCECD). Uma formulação matemática e um novo algoritmo BP são introduzidos para solucionar o PCECD.

O segundo problema tratado surge dos sistemas *multi-elos*. O Problema de Roteamento de Veículos Capacitado com 2 Elos (PRVC-2E) se caracteriza por definir o fluxo de cargas entre o depósito e seus consumidores finais, usando armazéns intermediários denominados satélites. Estes armazéns são responsáveis por integrar o primeiro *elo*, que contém o depósito, com o segundo *elo*, aonde estão dispostos os consumidores. Além do mais, as mercadorias podem ser consolidadas nos satélites a fim de minimizar os custos de entrega. São apresentados um modelo matemático para o PRVC-2E, que possui um número exponencial de variáveis, além de quatro famílias de desigualdades válidas para fortalecê-lo. A solução deste modelo, incluindo a separação das desigualdades válidas, é feita por um algoritmo Branch-and-cut-and-price (BCP). Os resultados obtidos pelo BCP mostram que o seu desempenho domina o de outros algoritmos propostos para o

PRVC-2E na literatura.

Palavras-chave: Problemas de Roteamento de Veículos, Problemas de Distribuição, Geração de Colunas, Branch-and-price, Planos de corte.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Over the years, Operations Research tools have contributed significantly to reduce costs involved in the transportation of goods in logistic systems and to a better use of existing transportation infra-structure. Probably one of the first contributions in this domain is due to Dantzig and Ramser [1959]. In that reference, a model and a solution procedure were given to solve the problem of routing vehicles to gas stations, from a central gasoline depot. Since that work, the term Vehicle Routing Problem (VRP) became used to represent a whole class of problems that involve routing vehicles from a central depot, in order to deliver (collect) goods to (from) customers (suppliers).

During the last fifty years or so, a huge growth on the number of different VRPs tackled both by the research community and by practitioners was noticed (see Laporte [2009] for an updated survey). Most of them incorporate operational constraints initially ignored by the first VRPs studied.

In Figure 1.1 we depict a solution for a VRP containing 3 vehicles and 10 customers. Each vehicle must leave the depot to visit a set of customers and return to the depot at end of the route. Each customer has a given demand and must be visited only once to be satisfied. In the figure, vehicles and customers are labeled as $k_1, \ldots, k_3$ and $c_1, \ldots, c_{10}$, respectively.

The VRP belongs to the class of NP-Hard problems, since the Traveling Salesman Problem (TSP) can be reduced to a VRP with a single uncapacitated vehicle. The need for better solutions for VRP and its variants motivated the development of an impressive number of exact and heuristic algorithms over the past decades. As the computational power available increased and the solution techniques improved, more real world applications have significantly shown to benefit from such developments. As a consequence, practitioners and researchers started to include more sophisticated constraints on VRPs aiming to solve real instances cases. Such modifications usually

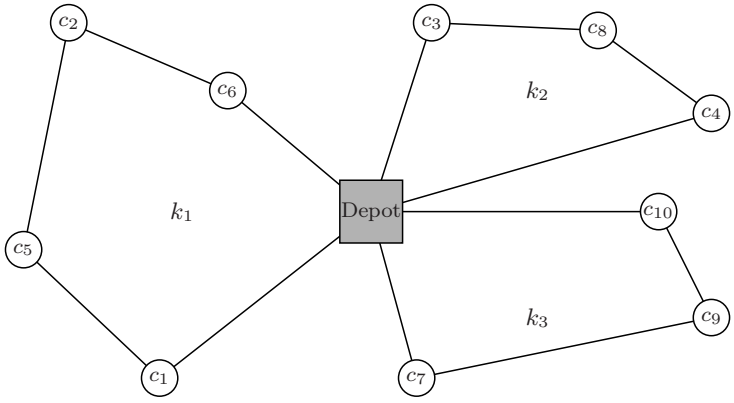


Figure 1.1: Solution example for a VRP with 10 customers using 3 vehicles.

lead to VRPs extensions for which to find optimal solutions become more complex, due to the increasing on the number of rules and constraints to tackle.

The recent advances in distribution systems have also motivated novel studies on VRPs. Logistics companies are extending the capabilities of warehouses for not only to store loads, but also allowing them to organize loads in order to improve their distribution along the logistic system. An example of such advances is the Cross-Docking warehouse [Apte and Viswanathan, 2000].

Cross-Docking (CD) is a warehouse proposed for helping in the transportation of loads without long-term inventory. Items delivered to a CD by inbound vehicles are immediately sorted out, reorganized based on customer demands and loaded into outbound vehicles for delivery, without the items being actually held in inventory at the warehouse. If any item is held in storage, it is usually for a brief period of time that is generally less than 24 hours [Yu and Egbelu, 2008].

The integration of VRPs with loads organization at CDs arises novel optimization problems. On the one hand, including scheduling decisions of loads organization at the CD, we enlarge the solutions space of problems. On the other hand, additional constraints must be introduced to ensure feasible solutions, defining more complex and challenging problems. The problem resulting from the integration of routing decisions on VRPs with loads organization at CDs is named Vehicle Routing Problem with Cross-Docking (VRPCD) [Lee et al., 2006].

On the VRPCD a set of unsplittable loads must be shipped from the suppliers to the respective customers. A fleet of vehicles is in charge of collecting the loads on suppliers and delivering them to the respective customers. Before to deliver products to customers, vehicles must stop at the CD to organize loads, allowing them to change

from/to vehicles, and eventually optimizing their delivery. As VRPs, different variants of VRPCD can be considered, according to their characteristics. One can, for example, impose time windows constraints on suppliers and customers to ensure them to be served by vehicles in a proper interval or can either add additional costs to be incurred when loads change from/to vehicles at the CD.

The definition of VRPCD imposes that a given load must be collected at its supplier and stop at the CD for consolidation before to be delivered to their respective customer. Such a constraint may become solutions more expensive. For that reason, we propose here to allow loads to be delivered bypassing the CD, whenever such an operation reduce costs. To that end, we introduce pickup and delivery routes on the VRPCD. Vehicles implementing such routes are responsible for collecting a set of loads and delivering them immediately later, without stopping at the CD. As a result of such a modification, we state a novel optimization problem named the Pickup and Delivery Problem with Cross-Docking (PDPCD).

Another warehousing strategy frequently used in distribution systems is found in multi-echelon systems, on which echelons are linked using warehouses named satellites. Loads from inbound vehicles of a given echelon are delivered to a satellite, which also allows loads to be organized before to be loaded into outbound vehicles and shipped to the another echelon. Satellites are similar to CDs. However, loads are obligated to change from vehicles on satellites, while the loads changing is not compulsory at the CD and occurs only whether it optimizes the delivery.

Different combinatorial optimization problems may arise by integrating VRPs and organization of loads at satellites. The most recent problem dealing with VRP and loads scheduling at satellites is the Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP) [Perboli et al., 2008b]. The 2E-CVRP is the basic version of multi-echelon systems, since only two echelons are considered. On the 2E-CVRP a given load is available on the depot and must be shipped to customers using the satellites as intermediate depots. The routing is partitioned into level 1, where trucks are in charge of shipping loads from the depot to satellites and level 2, where small vehicles deliver the loads from the satellites to customers. After level 1 routing takes place, the loads are unloaded from trucks and organized into small vehicles to follow in level 2 routing. Therefore, the 2E-CVRP consists of assigning routes for vehicles on level 1 and level 2 for shipping loads from the depot to customers, as well as to define the organization of loads at satellites.

Other approaches integrating VRPs with other logistics problems were massively studied over the last decades in the literature [Geoffrion et al., 1982]. The Inventory Routing Problem (IRP) [Bell et al., 83; Golden et al., 1984; Campbell et al., 2002;

Moin and Salhi, 2007] is a well-known approach that integrates VRP and the inventory control problem. Solutions for IRPs consist of finding routes for vehicles to supply customers' demands in a given planning horizon in order to minimize inventory and routing costs. Production planning problems and VRPs were also attacked together by different authors [Mak and Wong, 1995; Fumero and Vercellis, 1999; Yung et al., 2006; Lejeune and Ruszczynski, 2007]. Their solutions concern routes for vehicles satisfying constraints on lot sizing and input/output of materials, commonly found in supply chains.

Even though VRPs may be integrated with other logistics facilities, we focus on this thesis our research on the VRPCD, PDPCD and 2E-CVRP. Such problems were considered because they integrate routing decisions with loads organization at warehouses. Moreover, they are recent in the literature. Therefore, studies on VRPCD, PDPCD and 2E-CVRP can contribute with the continuous development of this promising area.

1.1 Objectives

Cross-Docking warehouses are widely used by global market companies like Toyota [Ohlmann et al., 2008], UPS [Forger, 1995] and Wal Mart [Gue, 2001]. However, the VRPCD literature has a small number of contributions [Lee et al., 2006; Wen et al., 2009; Musa et al., 2010; Liao et al., 2010; Tarantilis, 2012]. In addition, to the best of our knowledge, no exact solution approaches were reported for neither VRPCD or PDPCD.

Similarly to VRPCD and PDPCD, the 2E-CVRP is a quite recent problem in the literature. Although multi-echelon systems and their related problems are well known in the literature, the first to propose models and algorithms to integrate the consolidation decisions on satellites with routing decision was Perboli et al. [2008b]. After that, a few papers reported works on 2E-CVRP [Perboli and Tadei, 2010; Perboli et al., 2011; Crainic et al., 2011], but only one exact approach was proposed with a branch-and-cut algorithm.

The objective of this thesis is to investigate models and algorithms to solve VRPCD, PDPCD and 2E-CVRP. Specially, we are interested in exact algorithms based on decomposition approaches. Using such algorithms we expect to evaluate solutions faster for the respective problems. Because no results were reported in the literature using branch-and-price algorithms for VRPCD, PDPCD and 2E-CVRP, we investigate here to solve such problems using branch-and-price. For the 2E-CVRP we go

beyond and also investigate families of valid inequalities in a branch-and-cut-and-price framework.

1.2 Main Contributions

We are going to report along this thesis some contributions concerning integer programming modeling and solutions for VRPCD, PDPCD and 2E-CVRP. The major contributions are:

- Two Integer Programming Formulations for the VRPCD and associated branch-and-price algorithms for solving such formulations;
- A deeper study on algorithms for solving the resulting column generation subproblems. For instance, we implemented Dynamic Programming, Branch-and-cut and Heuristic algorithms for solving pricing problems;
- An Integer Programming Formulation and a branch-and-price algorithm to solve the PDPCD;
- An Integer Programming Formulation and valid inequalities are also proposed for the 2E-CVRP. A resulting branch-and-cut-and-price is also implemented to solve such formulation. Promising results are reported, including new optimality certificates for several instances of the literature.

In addition, we also contribute with scientific papers that have already been published (or accepted for publication):

- Santos F. A., da Cunha A. S., Mateus G. R., "Modelos de otimização para o Problema de Roteamento de Veculos com *Cross-Docking*", Simpósio Brasileiro de Pesquisa Operacional (SBPO), 2010 (in portuguese) [Santos et al., 2010]
- Santos F. A., da Cunha A. S., Mateus G. R., "Um Algoritmo Branch-and-price para o Problema de Roteamento de Veículos com *Cross-Docking* para Frotas Heterogêneas", Simpósio Brasileiro de Pesquisa Operacional (SBPO), 2011 (in portuguese) [Santos et al., 2011a]
- Santos F. A., da Cunha A. S., Mateus G. R., "A Branch-and-price algorithm for a Vehicle Routing Problem with Cross-Docking", Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS), 2011 [Santos et al., 2011c]

- Santos F. A., da Cunha A. S., Mateus G. R., "A novel column generation algorithm for the Vehicle Routing Problem with Cross-Docking", International Network Optimization Conference (INOC), 2011 [Santos et al., 2011b]

- Santos F. A., da Cunha A. S., Mateus G. R., "Branch-and-Price Algorithms for the Two-Echelon Capacitated Vehicle Routing Problem", Optimization Letters, 2012 [Santos et al., 2012a]

- Santos F. A., da Cunha A. S., Mateus G. R., "The Pickup and Delivery Problem with Cross-Docking", Computers and Operations Research, 2012 [Santos et al., 2012b]

1.3 Thesis Outline

On chapter 2 we discuss the background and related works associated with this thesis. We start introducing the classical VRP and some of its common variants: the VRP with Backhauls, the VRP with Time-Windows and the VRP with Pickup and Delivery. Then, we formalize the VRPCD, the PDPCD and the 2E-CVRP and present a literature review with the most relevant contributions for each problem.

Those models and algorithms used to solve the VRPCD are introduced on chapter 3. We propose three integer programming models for solving the VRPCD: one based on network flow formulation and the two others based on set partitioning formulation, leading to models with an exponential number of variables. We implemented branch-and-price algorithms to solve the set partitioning based models. The algorithm implementation details, including different approaches for solving the pricing problems are also discussed on chapter 3, where we also present the computational experience of solving the VRPCD using the three proposed models.

We propose on chapter 4 a set partitioning based formulation for the PDPCD and also implemented a branch-and-price algorithm to solve it. Modeling and algorithm implementation details as well as further computational results for the PDPCD are also discussed on this chapter.

On chapter 5 we present an integer programming model for 2E-CVRP which holds an exponential number of variables and constraints. In addition, we also investigate valid inequalities to strengthen such a model. A branch-and-cut-and-price algorithm is proposed to evaluate such a model. We describe the algorithm's implementation and report on computational results, comparing our method with previous exact solution approaches of the literature.

Finally, we conclude this thesis on chapter 6 and also discuss further steps of research.

Chapter 2

Background and Literature Review

The VRP is one of the most studied combinatorial optimization problems [Laporte, 2009; Toth and Vigo, 2002b]. As increased the computational resources over the years, allowing powerful software-based engines to solve VRPs, as became complex the problems, dealing with even more sophisticated real-world constraints. As a consequence of such a development, different VRP variants were proposed, including particular constraints from the real-world applications.

Recently, authors have proposed to integrate VRPs with logistics and distribution problems, leading to even more complex problems. In particular, the Vehicle Routing Problem with Cross-Docking (VRPCD), the Pickup and Delivery Problem with Cross-Docking (PDPCD) and the Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP) were proposed to deal with routing decisions integrated with consolidation of loads in specific warehouses, respectively named Cross-Docking and Satellites. Consolidation of loads is a modern trend in distribution systems, allowing loads to change from/to vehicles in order to satisfy routing constraints and minimizing costs.

In this chapter we discuss the classical VRP and its most common variants on section 2.1, including a literature review on the most relevant research conducted in this area. On sections 2.2, 2.3 and 2.4 we formalize the problems dealt in this thesis: the VRPCD, the PDPCD and the 2E-CVRP, respectively. A literature review concerning such problems is also provided on each section.

2.1 The Classical Vehicle Routing Problem

In this section we are going to introduce some basic definitions about VRP and its most common variants. We also introduce integer programming models and a literature review on the most effective solution methods for each problem. For further reading

on VRP and variants we refer to Toth and Vigo [2002b].

Let $G = (V, A)$ be a directed graph where $V = \{0, 1, \ldots, n\}$ represents the set of vertices and $A = \{(i, j) : i, j \in V; i \neq j\}$ denotes the set of arcs of G. Vertices $1, \ldots, n$ correspond to the customers of the network, whereas vertex 0 identifies the depot (or warehouse). For each arc $(i, j) \in A$ we associate a nonnegative cost c_{ij} to denote the cost to travel from vertex i to vertex j. Assume we are given a set $\mathcal{K}$, containing K vehicles, in charge to leave the depot for visiting customers and return back to depot at the end of the route. The VRP consists of assigning K routes, one for each vehicle of $\mathcal{K}$, for spanning all customers of G, minimizing the traveling costs, given by the sum of arcs costs traversed by vehicles.

Further on the above definition, generally authors include capacity constraints in the VRP, leading to the Capacitated VRP (CVPR). In the CVRP, we assume that a nonnegative demand q_i is assigned to each customer of G and consider a capacity limit Q^k for each vehicle $k \in \mathcal{K}$. Whether vehicles of set $\mathcal{K}$ hold the same capacity, the problem is called homogeneous and we simplify the capacity limit to Q, otherwise we call it as heterogeneous CVRP. Along this thesis, we assume homogeneous problems as default. We also make use of both terms the VRP and the CVRP to refer to the Capacitated VRP. Furthermore, we interchangeably use the terms loads, goods, products and demands as well as vehicles and routes.

Although we stated above that exactly K vehicles must be used in VRP solutions, is also possible to impose that feasible solutions use at most K vehicles. In addition, different objective functions could also be considered, for example, to minimize the number of vehicles used to meet the customers requirements, instead of minimizing the routing costs. All of these approaches are well-known in the literature and their use depend of the costs involved. In this thesis we deal only with the minimization of routing costs.

After introduced by Dantzig and Ramser [1959], an impressive number of algorithms were proposed to solve the VRP. A distinguished approach was proposed by Christofides et al. [1981], which uses a Lagrangean bound with minimum q-routes subproblems. They also describe a lower bound based on k-degree center trees which are minimum spanning trees having degree $K \leq d \leq 2K$ at the depot. Similar approaches also use K-trees and Lagrangean relaxation to derive strong lower bounds for the CVRP [Martinhon et al., 2004; Fisher, 1994]. Another family of exact algorithms to solve the CVRP is based on set partitioning formulation. Balinkski and Quandt [1964] were one of the first to use such an approach, after that different algorithms were proposed [Agarwal et al., 1989; Hadjiconstantinou et al., 1995]. The best results for the CVRP in the literature are reported by Fukasawa et al. [2006]. Their method works over the

intersection of two polytopes, one associated with a Lagrangean relaxation, the other defined by bound, degree and capacity constraints, leading to a problem with an exponential number of variables and constraints. The authors are able to solve instances of the literature with up to 135 customers. We refer to Toth and Vigo [2002a] for a survey on exact algorithms for the CVRP.

In the following we present a linear integer programming formulation for the CVRP extracted from Toth and Vigo [2002b]. The model is a *two-index flow formulation* that makes use of $O(n^2)$ binary decision variables x_{ij} indicating whether arc $(i, j) \in A$ is traversed in the solution or not, assuming respectively values 1 or 0.

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

$$\sum_{j\in V} x_{0j} = K \tag{2.2}$$

$$\sum_{i\in V} x_{i0} = K \tag{2.3}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.4}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.5}$$

$$\sum_{i\notin S} \sum_{j\in S} x_{ij} \geq \pi(S) \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{2.7}$$

The objective function (2.1) minimizes the piecewise costs of the traversed arcs in the solution. Constraints (2.2) and (2.3) impose that exactly K vehicles are used in the solution for visiting customers, while constraints (2.4) and (2.5) are respectively the *outdegree* and *indegree* constraints and are proposed to ensure that each customer is served exactly once in the solution.

Constraints (2.6) are the so-called capacity-cutset constraints. They are used to guarantee both connectivity and capacity constraint feasibility on solutions. The parameter $\pi(S)$ define the minimum number of vehicles needed to satisfy the demands of vertices of the set S. One can solve a bin-packing problem to evaluate $\pi(S)$ or alternatively can use the valid lower bound $\left\lceil \frac{\sum_{i\in S} q_i}{Q} \right\rceil$ [Cornuéjols and Harche, 1993]. In addition, the capacity-cutset constraints ensure subtour elimination on solutions, enforcing at least $\pi(S)$ vehicles to across a subset of vertices S.

It is easy to see that model (2.1)-(2.7) has an exponential number of constraints, due to (2.6). Therefore, to evaluate the Linear Programming (LP) bound for larger instances one needs to make use of decomposition methods. In this case, constraints 2.6 may be relaxed using a Lagrangean structure [Christofides et al., 1981; Fisher, 1994; Miller, 1995] or a Branch-and-cut algorithm [Araque et al., 1994; Achuthan et al., 2003] may be implemented, adding such constraints by demand on the model, until no more constraints can be found to improve the LP bound value.

Alternatively, is possible to overcome the drawback imposed by the exponential number of constraints (2.6) replacing them by the Miller-Tucker-Zemlin constraints [Miller et al., 1960], proposed for the Traveling Salesman Problem. These constraints make use of a new set of continuous variables $u_i : \forall i \in V \setminus \{0\}$ proposed to indicate the total load of the vehicle after serving customer i. Two new families of inequalities are created

$$u_i - u_j + Qx_{ij} \leq Q - q_j \qquad \forall i, j \in V \setminus \{0\}, i \neq j \tag{2.8}$$

$$q_i \leq u_i \leq Q \qquad \forall i \in V \setminus \{0\} \tag{2.9}$$

The model (2.1)-(2.5), (2.7)-(2.9) has $O(n^2)$ constraints and also defines a valid formulation for CVRP. In other words, the Miller, Tucker and Zemlin constraints state that, whether an arc (i, j) is selected for the solution ($x_{ij} = 1$), the inequality $u_i \leq u_j - q_j$ holds. Otherwise, whether $x_{ij} = 0$, the constraints are trivially satisfied. In addition, the total load of any route must satisfy the capacity constraint.

Although the LP bound of model (2.1)-(2.5), (2.7)-(2.9) can be easily evaluated without decomposition approaches, it is usually weaker than that bound provided by model (2.1)-(2.7). Some improvements and extensions were proposed to tightening such a bound [Desrochers and Laporte, 1991], but methods based on decomposition approaches are those providing the best results for VRP and its variants.

Two-index formulations are useful for modeling VRPs and its basic variants. However, they are not adequate for modeling complex VRP variants. For instance, in the VRP with Pickup and Delivery there are precedence constraints to be applied on a set of vertices that could not be modeled using a two-index formulation. For these situations, we must formulate the problem using three-index formulations.

A three-index formulation is obtained by replacing variables x_{ij} for variables $x_{ij}^k : \forall k \in \mathcal{K}$, to assume values 1 or 0 whether vehicle k traverses or not arc (i, j) on the solution, respectively. In addition, a new set of variables y_i^k may be used to represent whether customer i is visited by vehicle k, assuming value 1, or 0 otherwise. A linear

integer programming for the CVRP based on three-index formulation [Toth and Vigo, 2002b] follows.

$$\min \sum_{(i,j)\in A} c_{ij} \sum_{k\in\mathcal{K}} x_{ij}^k \tag{2.10}$$

$$\sum_{k\in\mathcal{K}} y_0^k = K \tag{2.11}$$

$$\sum_{k\in\mathcal{K}} y_i^k = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.12}$$

$$\sum_{j\in V} x_{ij}^k = \sum_{j\in V} x_{ji}^k = y_i^k \qquad \forall i \in V,\ \forall k \in \mathcal{K} \tag{2.13}$$

$$\sum_{i\in V} q_i y_i^k \leq Q \qquad \forall k \in \mathcal{K} \tag{2.14}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ij}^k \geq y_h^k \qquad S \subseteq V \setminus \{0\} : h \in S, \forall k \in \mathcal{K} \tag{2.15}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in \mathcal{K} \tag{2.16}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in V, \forall k \in \mathcal{K} \tag{2.17}$$

Constraint (2.11) ensures that all vehicles of set $\mathcal{K}$ are used in the solution, while constraints (2.12) enforce each customer to be served once for exactly one vehicle. The flow conservation constraints are expressed on (2.13), coupling arc and vertex variables for each vehicle in order to ensure feasibility on solutions. Capacity constraints are given on constraints (2.14). Finally, constraints (2.15) are so-called *edge-cuts* constraints and impose connectivity on solutions. Alternatively, such constraints may be replaced by the well-known *Subtour Elimination Constraints* (SECs) [Fisher and Jaikumar, 1981]

$$\sum_{i\in S} \sum_{j\in S} x_{ij}^k \leq |S| - 1 \qquad S \subseteq V \setminus \{0\} : |S| \geq 2, \forall k \in \mathcal{K}, \tag{2.18}$$

which ensures that a vehicle performs a path (instead of a cycle) spanning vertices of a given subset S. Once again, connectivity constraints (2.15) or (2.18) may be replaced by Miller, Tucker and Zemlin constraints in order to obtain a compact formulation, containing a polynomial number of variables and constraints. To that end, we introduce a set of continuous variables indexed by vertices and vehicles u_i^k and formulate the following constraints

$$u_i^k - u_j^k + Qx_{ij}^k \leq Q - q_j \qquad \forall i, j \in V \setminus \{0\} : i \neq j, \forall k \in \mathcal{K}, \tag{2.19}$$

$$q_i \leq u_i^k \leq Q \qquad \forall i \in V \setminus \{0\}, \forall k \in \mathcal{K}. \tag{2.20}$$

Regardless how connectivity and capacity constraints are defined, models based on three-index formulation are generally harder to solve than those based on two-index formulation, since both variables and constraints must be indexed by vehicles, becoming the solution space larger. For this reason, two-index formulations are preferred for modeling VRPs than three-index formulations. However, three-index models are required for modeling some VRP variants. In the next three sections, we discuss well-known variants of the VRP: the VRP with Backhauls on section 2.1.1, the VRP with Time Windows on section 2.1.2 and the VRP with Pickup and Delivery on section 2.1.3.

2.1.1 The Vehicle Routing Problem with Backhauls

The VRP with Backhauls (VRPB), also known as linehaul-backhaul VRP, extends the CVRP by partitioning the set of customers into two subsets: one composed by customers with a given demand of loads to be delivered (linehaul) and another of suppliers with loads to be collected (backhaul). This scenario arises frequently in supply chains environments, where suppliers and customers play different roles, but all of them must be visited to meet system requirements.

The VRPB is defined in a graph $G = (V, A)$ such as $V = \{0\} \cup L \cup B$ and $A = \{(i, j) : i, j \in V\}$. Sets $L = \{1, \ldots, n\}$ and $B = \{n + 1, \ldots, n + m\}$ contain respectively the linehaul and backhaul vertices. The vertex 0 is the depot of the problem, where a fleet of K vehicles is available. Thus, each route must start and finish at the depot. For each customer $i \in L \cup B$ we associate a demand $q_i > 0$ to be delivered or collected. The VRPB consists in assigning routes for vehicles to visit exactly once each customer, including the additional constraints: (i) they must leave the depot for visiting linehaul and/or backhaul customers; (ii) the sum of total demands for linehaul and backhaul customers (separately) must not exceed the vehicle's capacity and (iii) the linehaul customers must be visited before the backhaul (if any) in the route. Note that, the VRPB is reduced to the CVRP whether $B = \emptyset$. The objective is also to minimize routing costs.

We reproduce here the linear integer programming model for the VRPB proposed by Toth and Vigo [1997]. For such an expedite we modify the arcs set defined in section

2.1 to $\overline{A} = A_1 \cup A_2 \cup A_3$ and formulate the problem on a directed graph $\overline{G} = (V, \overline{A})$ where

$$A_1 = \{(i,j) \in A : i \in L \cup \{0\}, j \in L\}$$
$$A_2 = \{(i,j) \in A : i \in L, j \in B \cup \{0\}\}$$
$$A_3 = \{(i,j) \in A : i \in B, j \in B \cup \{0\}\}$$

Note that, the set of arcs $\overline{A}$ is a proper subset of A and contain only those feasible arcs for the VRPB. Those arcs connecting backhaul to linehaul customers are not in $\overline{A}$ as well as those arcs from the depot to backhaul customers. Thus, routes containing only backhaul customers are not allowed. We also define sets $\Delta_i^+ = \{j \in V : (i,j) \in \overline{A}\}$ and $\Delta_i^- = \{j \in V : (j,i) \in \overline{A}\}$ containing respectively the forwarding and the backwarding adjacent vertices of i on $\overline{G}$. The model is achieved by introducing arc variables $x_{ij} : (i,j) \in A$ for modeling whether a given network arc is used or not, assuming respectively values 1 or 0. A linear integer programming formulation for VRPB [Toth and Vigo, 2002b] follows.

$$\min \sum_{(i,j)\in\overline{A}} x_{ij} c_{ij} \tag{2.21}$$

$$\sum_{j\in\Delta_0^+} x_{0j} = K \tag{2.22}$$

$$\sum_{i\in\Delta_0^-} x_{i0} = K \tag{2.23}$$

$$\sum_{j\in\Delta_i^+} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.24}$$

$$\sum_{j\in\Delta_i^-} x_{ji} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.25}$$

$$\sum_{i\notin S} \sum_{j\in\Delta_i^+} x_{ij} \geq \pi(S) \qquad S \subseteq L, S \neq \emptyset \tag{2.26}$$

$$\sum_{i\notin S} \sum_{j\in\Delta_i^+} x_{ij} \geq \pi(S) \qquad S \subseteq B, S \neq \emptyset \tag{2.27}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \overline{A} \tag{2.28}$$

Similarly to the previous VRP models discussed here, the objective function (2.21) is to minimize the sum of the costs of the traversed arcs in the solution. Constraints (2.22)-(2.25) ensure all vehicles on solutions and that each customer is served exactly once. Again, the capacity-cutset constraints enforce connectivity, subtour elimination

and capacity feasibility on solutions. However, these constraints are stated now independently for linehaul and backhaul customers, respectively on (2.26) and (2.27). Over again, the parameter $\pi(S)$ indicates the minimum number of vehicles (or a proper lower bound) needed to meet all demands of customers of a given subset S.

Deif and Bodin [1984] were the first to introduce the VRPB. They proposed an extension of the well-known Clarke and Wright algorithm for VRP [Clarke and Wright, 1964]. After that, different authors reported works for solving the VRPB and small variants. Goetschalckx and Jacobs-Blecha [1989, 1993] proposed a non-linear mathematical formulation for VRPB and also presented exact and heuristic algorithms to solve it. Toth and Vigo [1997, 1999] introduced a linear integer programming formulation for VRPB. They also implemented branch-and-bound and cutting plane algorithms for solving such a model. An exact approach for VRPB based on set partitioning formulation and column generation was introduced by Mingozzi et al. [1999]. The authors solved the pricing problem using both exact and heuristic algorithms. Small variants of VRPB, including those with time windows constraints, are also concerned by different authors [Anily, 1996; Duhamel et al., 1997; Thangiah et al., 1996].

2.1.2 The Vehicle Routing Problem with Time Windows

The VRP with Time Windows (VRPTW) extends the CVRP by introducing time windows constraints to limit the starting and ending time on which a customer is available to be visited by vehicles. A function to define the distance traveled by vehicles for each time unit is given. Generally, such a function is given in terms of constant values, but there are also works on the literature concerning more complex functions.

On the VRPTW, early services are not allowed. Whether a vehicle arrives at a customer before its time window interval, such a vehicle must wait the early time of the customer. On the opposite, a late service can be checked using two different approaches. On the one hand, the VRPTW with soft time windows allows an out to date service to be performed under a given penalty. On the other hand, services out to date are not feasible on the VRPTW with hard time windows, which become the problem more complex. The VRPTW is a NP-Hard problem, since it extends the CVRP. Indeed, according to Savelsbergh [1985], even to find a feasible solution for a VRPTW with hard time windows is a NP-Hard problem. Because the problem with hard time windows have received attention from the most of work of the literature, it becomes the standard VRPTW problem. Therefore, hereinafter we use the term VRPTW to refer to the problem with hard time windows.

In what follows, we reproduce the linear integer programming formulation for

VRPTW from Toth and Vigo [2002b]. For each vertex $i \in V$ of graph $G = (V, A)$ we associate an interval $[a_i, b_i]$ containing respectively the early a_i and the due time b_i on which vertex i can be served. The earliest time a vehicle can leave the depot (E) and the latest time it can arrive (L) are used respectively to define the time windows for the depot $[a_0, b_0] := [E, L]$. In addition, the parameter t_{ij} define the time needed to across each arc $(i, j) \in A$ and the service time of each customer i is given as s_i. Such quantities allow us to execute pre-process routines to remove arcs from the graph. Whether for two any customers $i, j \in V$ the following inequality holds $a_i + s_i + t_{ij} > b_j$, we can drop the arc (i, j) from the graph, since the path containing customer j immediately later i is not feasible due to timing constraints.

Two types of decision variables are used for modeling the VRPTW: binary arc variables x_{ij}^k for modeling whether vehicle k across the arc (i, j) on the solution or not, assuming respectively values 1 or 0 and continuous time variables w_i^k to specify the start of service on vertex i, when performed by vehicle k.

$$\min \sum_{(i,j)\in A} c_{ij} \sum_{k\in\mathcal{K}} x_{ij}^k \tag{2.29}$$

$$\sum_{j\in\Delta_0^+} x_{0j}^k = 1 \qquad \forall k \in \mathcal{K} \tag{2.30}$$

$$\sum_{k\in\mathcal{K}} \sum_{j\in\Delta_i^+} x_{ij}^k = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.31}$$

$$\sum_{j\in\Delta_i^-} x_{ji}^k - \sum_{j\in\Delta_i^+} x_{ij}^k = 0 \qquad \forall k \in \mathcal{K},\ \forall i \in V \tag{2.32}$$

$$w_i^k + s_i + t_{ij} - w_j^k \leq M_{ij}(1 - x_{ij}^k) \qquad \forall k \in \mathcal{K},\ \forall (i,j) \in A \tag{2.33}$$

$$a_i \sum_{h\in\Delta_i^-} x_{hi}^k \leq w_i^k \leq b_i \sum_{j\in\Delta_i^+} x_{ij}^k \qquad \forall k \in \mathcal{K},\ \forall i \in V \tag{2.34}$$

$$\sum_{i\in V} q_i \sum_{j\in\Delta_i^+} x_{ij}^k \leq Q \qquad \forall k \in \mathcal{K} \tag{2.35}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall (i,j) \in A, \forall k \in \mathcal{K} \tag{2.36}$$

The sum of routing costs is minimized in the objective function (2.29). Constraints (2.30) enforces all vehicles to be used in the solution, while the flow conservation constraints are stated on (2.31) and (2.32). Timing constraints are imposed on inequalities (2.33) and (2.34). The constants M_{ij} on constraints (2.33) must be large enough to ensure feasible solutions. It can be evaluated as $M_{ij} := b_i + s_i + t_{ij}, (i, j) \in A$.

Indeed, due to constraints (2.34) variables $w_i^k := 0$ whether vehicle k does not visit customer i in the solution. Finally, the capacity constraints for each vehicle are expressed on (2.35) and (2.36) define the binary decision variables space.

Pullen and Webb [1967] and Knight and Hofer [1968] were the pioneering to report works on the VRPTW. Due to the NP-Hardness of the VRPTW, the most of solutions are concerned on heuristic approaches. The first methods were proposed to perform route construction and route improvement routines in order to achieve good quality local optima solutions [Solomon, 1986, 1987; Potvin and Rousseau, 1993]. Some methods include preprocessing routines on the graph to reduce the solutions space in order to facilitate the search [Desrosiers et al., 1995]. Recent methods are based on modern meta-heuristics, including Tabu Search [Taillard et al., 1997; Chiang and Russell, 1997; Potvin et al., 1996], Simulated Annealing [Chiang and Russell, 1996; Osman, 1993] and Evolutionary methods [Potvin and Bengio, 1996; Homberger and Gehring, 1999; Alvarenga et al., 2007]. The best upper bounds reported for VRPTW were achieved using one of these modern methods or by mixing their characteristics on hybrid approaches.

As upper bounds, different works were also proposed to derive lower bounds for the VRPTW, most of them dealing with decomposition approaches. Lagrangian relaxation approaches were widely studied for the VRPTW [Christofides et al., 1981; Kohl and Madsen, 1997; Kallehauge et al., 2006]. Different structures can be exploited as Lagrangian subproblem. One of the most successful approaches consists of relaxing the constraints (2.31) of model (2.29)-(2.36), obtaining an elementary restricted shortest path problem as Lagrangian subproblem. The lower bound derived from this Lagrangian approach is equivalent of that from column generation methods based on set partitioning formulation [Desrochers et al., 1992; Kohl et al., 1999; Chen and Xu, 2006]. A shortcoming of such approaches is the complexity of the resulting subproblem, proven to be NP-Hard [Feillet et al., 2004]. Alternative approaches were proposed to relax the elementary condition of the subproblem obtaining a nonelementary restricted shortest path subproblem, which is solved using pseudo-polynomial algorithms. The best results reported in the VRPTW literature were achieved using such an approach.

2.1.3 The Vehicle Routing Problem with Pickup and Delivery

The VRP with Pickup and Delivery (VRPPD) arises when a fleet of vehicles must satisfy a set of transportation requests. Each request defines a pickup and a delivery point, where a given demand must be respectively collected and delivered. The VRPPD is a one-to-one problem, therefore each pickup point is associated to a delivery

point. The VRPPD extends the CVRP by partitioning the customer vertices into two subsets, one for pickup and another for delivery, and defining a bijection between them. Furthermore, precedence constraints are introduced on pickup vertices, which must be visited before their corresponding delivery vertices.

Small variants of the VRPPD can easily be found in the literature. The most common variants are the VRPPD with Time Windows (VRPPDTW) [Dumas et al., 1991] and the Dial-a-Ride Problem (DARP) [Cordeau and Laporte, 2007]. In the first, each pickup and delivery point can be visited only within a given interval where the service is available, while the second concerns specific constraints for transporting people instead of loads, as user inconvenience and maximum ride time restrictions for passengers. The VRPPD and its variants have a lot of practical applications, including the transport of disabled and elderly and a set of courier services. We refer to Berbeglia et al. [2007] for more variants and applications of VRPPD.

Let $G = (V, A)$ be a directed graph with the vertices set partitioned into $V = \{\{0\}, P, D\}$, where 0 is a depot, $P = \{1, 2, \ldots, n\}$ is a set of pickup points and $D = \{n+1, n+2, \ldots, 2n\}$ is a set of delivery points. The arcs set is defined as $A = \{(i, j) : i, j \in V\}$ and for each arc an associated cost c_{ij} is given. A set of transportation requests T comprises n triples $(i, n+i, q_i) : i = \{1, 2, \ldots, n\}$, each one with a load $q_i > 0$ to be shipped from $i \in P$ to $n+i \in D$ on graph G. We are given a set $\mathcal{K}$ with K vehicles with capacity Q, which is available at the depot. The VRPPD consists of assigning a set of K routes for vehicles satisfying all transportation requests minimizing routing costs, given by the sum of traversed arcs.

We present next an integer programming formulation for the VRPPD based on network flows. To that end, we include on the graph G an artificial copy of the depot $0'$, which is connected to all pickup and delivery vertices, and define a set of K commodities (one to represent each vehicle of $\mathcal{K}$) to be shipped along the network from 0 to $0'$. Three types of decision variables are used: f_{ij}^k represents the amount of flow passing through arc $(i, j) \in A$ for commodity k; x_{ij}^k determines whether arc (i, j) is used to ship commodity k ($x_{ij}^k = 1$) or not ($x_{ij}^k = 0$); and y_i^k is a binary variable used to define whether vertex $i \in P \cup D$ is activated (or not) by commodity k. The VRPPD model follows.

$$\min \sum_{k\in\mathcal{K}} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{2.37}$$

$$\sum_{j\in P} f_{0j}^k = \sum_{i\in P} y_i^k + \sum_{n+i\in D} y_{n+i}^k + 1 \qquad \forall\, k \in \mathcal{K} \tag{2.38}$$

$$\sum_{j\in\Gamma_i^+} f_{ij}^k - \sum_{h\in\Gamma_i^-} f_{hi}^k = -y_i^k \qquad \forall i \in P \cup D, \forall\, k \in \mathcal{K} \tag{2.39}$$

$$f_{ij}^k \leq |V| x_{ij}^k \qquad \forall (i,j) \in A, \forall\, k \in \mathcal{K} \tag{2.40}$$

$$\sum_{j\in\Gamma_i^+} x_{ij}^k + \sum_{h\in\Gamma_i^-} x_{hi}^k = 2y_i^k \qquad \forall i \in P \cup D, \forall\, k \in \mathcal{K} \tag{2.41}$$

$$\sum_{j\in P} x_{0j}^k = 1 \qquad \forall\, k \in \mathcal{K} \tag{2.42}$$

$$\sum_{k\in\mathcal{K}} y_i^k = 1 \qquad \forall i \in P \cup D \tag{2.43}$$

$$y_i^k - y_{n+i}^k = 0 \qquad \forall i \in P, \forall\, k \in \mathcal{K} \tag{2.44}$$

$$\sum_{j\in\Gamma_i^+} f_{ij}^k \geq \sum_{j\in\Gamma_{n+i}^+} f_{n+ij}^q \qquad \forall i \in P, \forall\, k \in \mathcal{K} \tag{2.45}$$

$$\sum_{i\in P} y_i^k q_i \leq Q \qquad \forall\, k \in \mathcal{K} \tag{2.46}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall\, k \in \mathcal{K} \tag{2.47}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in P \cup D, \forall\, k \in \mathcal{K} \tag{2.48}$$

Similarly to the CVRP, VRPB and VRPTW the objective function (2.37) is to minimize routing costs. On the constraints side, (2.38) define the amount of flow is provided by the depot for each commodity, (2.39) are the flow balancing equations, while (2.40) are coupling constraints used to link f_{ij}^k and x_{ij}^k variables. Together, constraints (2.41) and (2.42) ensure that each commodity flows along the network (likewise, each vehicle must be used in the routing solution). Constraints (2.43)-(2.45) enforces that only one commodity is used to satisfy demands of pickup and delivery points respecting the precedence constraints (the pickup point must be visited before the respective delivery point). Finally, (2.46) state the capacity constraints and (2.47)-(2.48) define the decision variables space.

The first works on VRPPD were proposed for dial-a-ride scenarios. The early study was reported by Wilson et al. [1971]. After that, a lot of works were conducted for VRPPD and its variants. Stein [1978a,b] proposed algorithms for solving respectively the DARP and the VRPPD with a single uncapacitated vehicle. A general study

on VRPPD and VRPPDTW is presented by Savelsbergh and Sol [1995], including solution approaches derived from exact and heuristic methods. The best results in the literature for the VRPPD were recently reported by Ropke and Cordeau [2009] and Baldacci et al. [2011] using branch-and-cut-and-price algorithms. A survey including a classification scheme of the most common pickup and delivery problems is provided by Berbeglia et al. [2007].

2.2 The Vehicle Routing Problem with Cross-Docking

As previously described in the above sections, the need for better solutions for the VRP and its variants motivated, over the past decades, the development of an impressive number of algorithms, both exact [Christofides et al., 1981; Fisher, 1994; Fischetti et al., 1994; Martinhon et al., 2004; Baldacci et al., 2004; Fukasawa et al., 2006; Baldacci et al., 2008] and heuristic [Gendreau et al., 1994, 1997; Golden et al., 1998; Alvarenga et al., 2007]. As the computational power available increased and the solution techniques improved, more real world applications have shown to significantly benefit from such developments. Practitioners and researchers started to integrate VRP with production planning or distribution systems, to tackle even more sophisticated logistic systems. Cross-Docking is one of such systems to which VRP has been integrated with.

Cross-Docking (CD) is a recent warehousing technology aimed to reduce inventory costs in supply chain systems [Apte and Viswanathan, 2000]. Goods collected by a set of inbound vehicles are delivered at the CD station and after the items are consolidated and grouped at the dock, they are moved to the vehicles responsible for delivering them to their final destinations. A CD can be seen as a warehouse where a very reduced amount of goods are kept in a short term inventory (generally less than 24 hours). No long term inventory is available at the docks.

Over the last decades, CD warehouses have been widely used in supply chains systems to connect suppliers and customers. Vehicles are in charge of collecting loads at suppliers and return to the CD, allowing the consolidation of loads. At consolidation step, the collected loads can be organized, moving from/to vehicles in order to optimize their delivery. After that, vehicles leave the CD to deliver loads to the respective customers. Global companies such as Toyota [Witt, 1998], Walmart [Gue, 2001] and UPS [Forger, 1995] reported works describing how CD systems implementation help their business.

In order to operate in such an expedite fashion, a CD system must deal appropriately with complex issues like, for example, how truck loading and unloading operations should be scheduled at the docks [Boysen, 2010] and how vehicles should be routed to collect and deliver the goods [Salani, 2005]. The way goods are collected and delivered is of crucial importance for determining the workload and the time needed to reorganize them at the docks. The more integrated the resolution of these two problems is, the more cost and time effective a cross-docking system should be. Bearing that in mind, a substantial amount of research was dedicated to propose ways to integrate the resolution of these problems. A detailed review of the papers dedicated to this matter could be found in Lee et al. [2006]; Wen et al. [2009]; Boysen and Fliedner [2010]; Santos et al. [2010, 2011c,b,a].

As a result of such attempts of integration, Lee et al. [2006] proposed the Vehicle Routing Problem with Cross-Docking (VRPCD). In that problem, a fleet of vehicles is in charge of collecting goods from suppliers, delivering them to their final destinations, after loading and unloading operations take place at the CD. The goods are collected and delivered considering time windows constraints. Each time a good is moved from/to a vehicle at the dock, an additional amount of time is needed to implement the operation. The goal in VRPCD is to find routes (satisfying vehicle's capacities and time windows on the nodes) such that all goods are collected and delivered to their final destinations and the total transportation cost is minimized.

Only a few works dealing with the VRPCD were reported in the literature. After introducing the VRPCD, Lee et al. [2006] presented a mathematical formulation for the problem and implemented a heuristic algorithm based on Tabu Search in order to obtain approximated solutions faster. Computational experiments were conducted using random generated instances with 10, 30 and 50 transportation requests and the results achieved in the worst case 5% from optimal values.

In Wen et al. [2009] the authors solved a similar VRPCD from that introduced by Lee et al. [2006]. However they relaxed the constraint imposed in Lee et al. [2006], on which all vehicles must be in the dock to perform the consolidation of loads. Another mathematical model based on three-index formulation was introduced by the authors. They also implemented a Tabu search heuristic to approximate solutions. Experimental results were conducted based on real data instances with up to 200 requests (400 vertices) and the solutions achieved with the heuristic algorithm were within 5% from lower bound values. Such lower bounds were evaluated by disregarding the time spent for loading/unloading products at the CD and solving separately the two resulting VRPTW for suppliers and customers using the algorithm of Kallehauge et al. [2006].

Further works may also be found in the VRPCD literature [Musa et al., 2010;

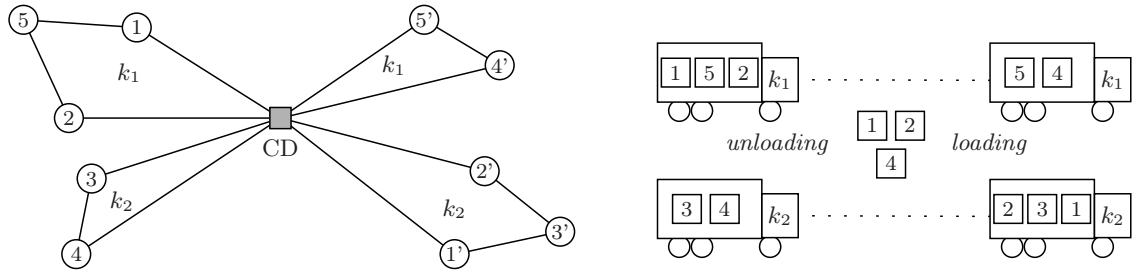
Liao et al., 2010; Tarantilis, 2012]. In general, they propose methods based on heuristic approaches aiming to improve results of the pioneering works of Lee et al. [2006] and Wen et al. [2009]. We recall that, no exact solution approaches could be found for VRPCD in the literature.

In this thesis, we consider a a slightly different VRPCD of that previously dealt in Lee et al. [2006]; Wen et al. [2009]. We neglect time windows constraints and introduce a cost to be incurred in the consolidation step whenever a good is moved from a vehicle to another at the CD. The cost for loading/unloading products at the CD is incurred because such operations requires human and/or machine efforts and should be managed in real applications of cross-docking systems. For solving such a problem we make use of exact solution approaches based on column generation scheme.

Let $G = (V, A)$ be a directed graph, where $V = \{\{0\}, S, C\}$, $S = \{1, \ldots, n\}$ is a set of suppliers, $C = \{1', \ldots, n'\}$ is a set of customers and 0 denotes the CD. Different suppliers may take the same geographical location, and customers as well. Assume that $A = A_S \cup A_C$ ($A_S \cap A_C = \emptyset$), where $A_S = \{(i, j) : i, j \in \{0, 1, \ldots, n\}\}$ denotes the set of arcs connecting the suppliers as well as the CD and $A_C = \{(i, j) : i, j \in \{0, 1', \ldots, n'\}\}$ denotes the set of arcs connecting the customers and the CD. Consider that $P = \{p_i := (i, i', q_i) : i = 1, \ldots, n\}$ denotes a set of transportation requests, each one given by an unsplittable load $q_i > 0$ to be shipped from a supplier i to a customer i'. Consider as well that a set $\mathcal{K}$ with K homogeneous vehicles of capacity Q is given to ship products along the network.

Let us assign costs $\{c_{ij} \geq 0 : (i, j) \in A\}$ (satisfying the triangle inequalities) to the arcs of G. The VRPCD consists of finding $2K$ routes to visit once suppliers and customers in order to respectively collect and deliver products, without exceeding the vehicles capacities. Each route must start and end at the CD. Furthermore, before delivering products to customers, each vehicle must necessarily stop at the CD, in order to allow loads to change vehicles and eventually minimize the delivery costs. Whenever a product p_i is loaded/unloaded from a vehicle to another at the CD, a cost $c_i \geq 0$ is incurred. Thus, the objective of VRPCD is to find routes such that the total cost (given by the transportation costs and the costs of changing loads at the CD) is minimized. In Figures 2.1a and 2.1b, we depict a set of routes for vehicles k_1 and k_2, where products p_1, p_2 changed from vehicle k_1 to k_2 and product p_4 from k_2 to k_1.

The costs incurred to change products at the CD define the VRPCD structure. On the one hand, suppose we assign costs for loading/unloading products at the CD as $c_i = 0$, $\forall p_i \in P$. The resulting problem can be decomposed into two independent CVRP: one considering the CD and suppliers and another for the CD and customers. On the other hand, if such costs are assigned as $c_i > U_i$, $\forall p_i \in P$, where U_i is a



(a) Vehicles routing (b) Loads change at CD

Figure 2.1: An example of a VRPCD solution concerning 5 products and 2 vehicles.

threshold cost to ensure that load q_i is not loaded/unloaded at the CD in the optimal solution, the VRPCD is reduced to the VRPPD, since each product is going to be collected and delivered using the same vehicle. In this paper we addressed the VRPCD concerning costs $0 < c_i < U_i : p_i \in P$, since solutions for CVRP and VRPPD are well known in the literature.

For that case on which the routing costs are neglected ($c_{ij} = 0,\ \forall\,(i,j) \in A$), the VRPCD is reduced to the truck scheduling problem. In this problem one deals only with the scheduling of loads at the CD, aiming to organize products from inbound trucks into outbound trucks in order to optimize Cross-Docking operations (cost/time). Different works addressed the truck scheduling problem, see Boysen [2010], Chen and Lee [2009] and Yu and Egbelu [2008] for details.

Let us now introduce a linear mixed integer programming model for VRPCD. The proposed model is based on a commodity flow formulation and uses K commodities, to be shipped along the network. In this model, each commodity is used to indicate a vehicle route. For this reason, hereinafter we interchangeably use the terms commodity and vehicle. For each supplier/customer we assign an unitary demand of commodity, which can be satisfied by any vehicle. Indeed, we include in the vertices set an artificial copy of the CD $0'$, which requires $2K$ units of commodities, K coming from suppliers and K from customers. On the arcs set we add the corresponding edges $(i, 0') : i \in S \cup C$ and its associated cost. Each commodity must be shipped along the network from 0 to $0'$ properly meeting the demand of suppliers/customers.

For modeling purposes, we use a set of variables f_{ij}^k to indicate the flow of commodity k on the arc (i, j). Binary arc variables x_{ij}^k control whether an arc (i, j) is traversed by vehicle k in the solution, assuming value 1 or 0 otherwise. Topological variables y_i^k assume value 1 if and only if the demand of vertex i is satisfied by commodity k. Finally, the binary variables u_i^k define for each request p_i if vehicle k satisfy the demands of supplier i and customer i' or not, assuming respectively values 1 or 0.

A linear mixed integer programming model for VRPCD is (2.49)-(2.60).

$$\min \sum_{(i,j)\in A} c_{ij} \sum_{k\in K} x_{ij}^k + \sum_{p_i\in P} c_i(1 - \sum_{k\in K} u_i^k) \tag{2.49}$$

$$\sum_{j\in S} f_{0j}^k = \sum_{i\in S} y_i^k + 1 \qquad \forall k \in K \tag{2.50}$$

$$\sum_{j\in C} f_{0j}^k = \sum_{i\in C} y_i^k + 1 \qquad \forall k \in K \tag{2.51}$$

$$\sum_{k\in K} y_i^k = 1 \qquad \forall i \in S \cup C \tag{2.52}$$

$$\sum_{j\in\Delta_i^+} f_{ij}^k - \sum_{j\in\Delta_i^-} f_{ji}^k = -y_i^k \qquad \forall i \in S \cup C, \forall k \in K \tag{2.53}$$

$$f_{ij}^k \leq |P| x_{ij}^k \qquad \forall (i,j) \in A, \forall k \in K \tag{2.54}$$

$$\sum_{j\in\Delta_i^+} x_{ij}^k + \sum_{j\in\Delta_i^-} x_{ji}^k = 2y_i^k \qquad \forall i \in S \cup C, \forall k \in K \tag{2.55}$$

$$\sum_{i\in S} y_i^k d_i \leq Q \qquad \forall k \in K \tag{2.56}$$

$$\sum_{i\in C} y_i^k d_i \leq Q \qquad \forall k \in K \tag{2.57}$$

$$y_i^k + y_{i'}^k - 1 \leq u_i^k \qquad \forall k \in K, \forall p_i \in P \tag{2.58}$$

$$y_i^k + y_{i'}^k \geq 2u_i^k \qquad \forall k \in K, \forall p_i \in P \tag{2.59}$$

$$x_{ij}^k, y_i^k, u_i^k \in \{0,1\} \tag{2.60}$$

The objective function (2.49) minimizes the sum of routing costs plus loading/unloading costs incurred at the CD in the consolidation step. Constraints (2.50) and (2.51) defines the quantity of flow for each commodity to satisfy suppliers/customers demands. Note that, the commodity flow corresponds to the sum of the demands for suppliers/customers plus a unitary demand for the artificial depot. Constraints (2.52) and (2.53) ensure each demand on suppliers/customers to be met and the balance of flow on each vertex. Coupling constraints on flow and arc variables are given in (2.54), while equalities (2.55) define the flow conservation constraints for arcs variables. The capacity constraints for both the suppliers and the customers are given respectively in inequalities (2.56) and (2.57), while constraints (2.58) and (2.59) control the variables of loading/unloading loads at the CD. Finally, the binary variables bounds are given in (2.60).

2.3 The Pickup and Delivery Problem with Cross-Docking

A common feature of all previous approaches that have dealt with VRPCD is the assumption that vehicles must stop at the CD after the requests are collected from suppliers. That applies even if the vehicle collects and delivers the same requests. Of course, allowing vehicles to avoid the stop at the CD in such cases may reduce the transportation costs while, at the same time, frees space and resources at the station.

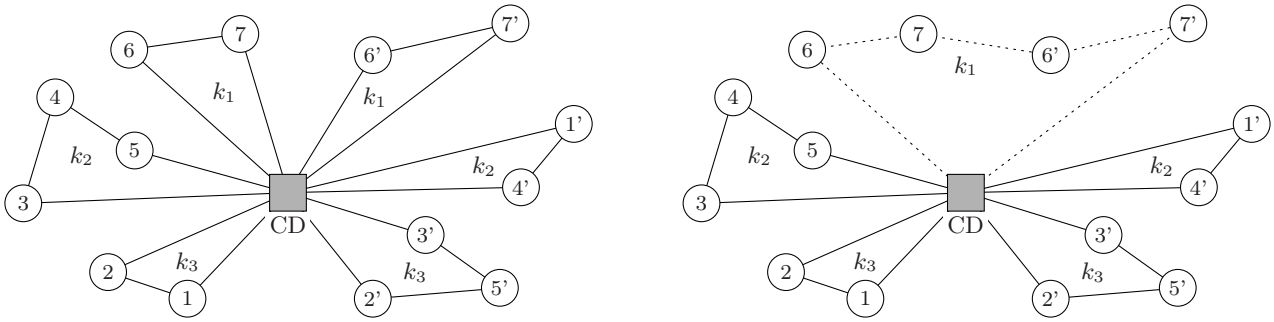
For that reason, we propose in this thesis to extend previous works on VRPCD and introduce a new problem, on which vehicles are allowed to deliver the requests immediately after collecting them, avoiding to stop at the CD for consolidation. The proposed approach considers two types of routes: pickup and delivery routes [Savelsbergh and Sol, 1995] (when the vehicle does not stop at the CD) and routes that stop at the CD to implement load changes. We name such a problem as the Pickup and Delivery Problem with Cross-Docking (PDPCD). Therefore, the proposed PDPCD is suitable to consider all the problems between a classical Pickup and Delivery Problem [Savelsbergh and Sol, 1995] and a classical VRPCD [Lee et al., 2006], whether all vehicles stop at the CD.

To define the PDPCD we modify the graph $G = (V, A)$ defined in section 2.2 by including the arcs $\{(i, j) : i \in S, j \in C\}$ on set A. Two types of routes are considered:

1. routes that start at the CD, visit a subset of suppliers, return to the CD, implement load changes at the CD, leave the CD to visit a subset of customers. After visiting the last customer, the vehicle returns empty to the CD. These routes either collect loads that they do not deliver, or deliver loads that they do not collect, or both. Such routes are named *docking routes*.

2. routes that start at the CD, visit a subset of suppliers and after the last one is visited, start delivering the collected loads to the customers, without a stop at the CD. Only after the last customer is visited, the vehicle returns empty to the CD. These are the *pickup and delivery routes*.

To further illustrate the differences between the two types of routes, in Figures 2.2a and 2.2b we depict two sets of 3 routes. In Figure 2.2a, only *docking routes* are used. In Figure 2.2b, routes of both types are considered. Note that, in Figure 2.2b, the route implemented by vehicle k_1 visits customer $6'$ right after collecting request p_7 at supplier 7. Therefore, vehicle k_1 delivers and collects the same set of products and does not stop at the CD.



(a) One possible solution for VRPCD

(b) One possible solution for PDPCD

Figure 2.2: Differences between possible solutions for VRPCD and PDPCD, with $\mathtt{K} = 3, n = 7$. In the figures, k_1, k_2 and k_3 denote routes.

Depending on the geographical distribution of suppliers and customers and on how loading/unloading costs compare to arc costs, optimal solutions to PDPCD may involve both types of routes or not. If loading/unloading operations are too costly, optimal PDPCD solutions are likely to include more *pickup and delivery routes*. On the contrary, if load changing costs are zero, *docking routes* should be selected more frequently.

Compared to other routing models that integrate cross-docking with vehicle routing in the literature, the introduction of *pickup and delivery routes* allowed substantial reductions in the transportation costs. In the worst case, solutions for the PDPCD have the same cost of those obtained for VRPCD (whether no pickup and delivery routes are used).

2.4 The Two-Echelon Capacitated Vehicle Routing Problem

Recently, different approaches have been proposed to predict and control the impact of the transportation of freights within city domains, in terms of traffic congestion, pollution and noise. An important contribution in this area is due to Crainic et al. [2009], which propose that the transportation of goods in urban areas has to be thought in an integrated way. What means that, instead of looking for solutions (routes, assignment of vehicles to routes, customers' visiting times) for one particular firm individually, one should look for solutions that treat each firm as a part or component of a whole *City Logistics system*, as defined by Crainic et al. [2009]. Operating under this new perspective, individual carriers may eventually share operations at depots, warehouses, cross-dockings, vehicles and other assets.

In turn, the design of integrated methods for the distribution of goods poses several new challenges, from an optimization perspective. For example, optimization models and algorithms have to explicitly take into account the possibility of consolidation of loads of several carriers in different warehouses and the coordination of shared freights in charge of deliverying goods to final customers.

One successful approach for the integrated transportation of goods in urban areas is the multi-echelon transportation system. It consists in using intermediate warehouses to consolidate the shipment of products from one or more depots or suppliers to final customers [Ricciardi et al., 2002; Topan et al., 2009]. In a multi-echelon system, the direct flow of goods from suppliers to the final destination is not allowed. Instead, factories, depots, warehouses and customers are organized in layers (or levels) and only the transportation of goods from/to entities in the same layer is allowed. One advantage of multi-echelon systems is that vehicles in each layer can be conveniently sized in order to satisfy regulations imposed on cargo transportation in specific urban areas.

Layers in multi-echelon systems are usually set according to the problem to be solved. The most common choice is the two-echelon system, where intermediate depots, named satellites, are placed between suppliers and final customers [Crainic et al., 2010; Jung and Mathur, 2009]. In such a system, goods are shipped (usually in large quantities) from the central depot (supplier) to the satellites, where the consolidation of goods takes place. That means that products are moved to smaller trucks in order to be delivered to their final destinations. The consolidation of goods at the satellites is a quite effective tool for improving efficiency of transportation systems in urban areas. One the one hand, it allows goods supplied by different providers to be transported together, to destinations that are either the same or that lie close to each other. On the other, it allows goods to be placed in vehicles of size and weight suitable to the regions these goods are going to be delivered.

Two-echelon transportation systems can be turned into cost effective city logistic systems if, for example, the underlying freight management and routing decisions are thought in an integrated way. Accordingly, researchers and practitioners have investigated ways to cope with these two problems into a single integrated Combinatorial Optimization Problem. Such integration attempts resulted in what is named as the 2E-CVRP [Perboli et al., 2008b; Perboli and Tadei, 2010; Perboli et al., 2011]. Solving 2E-CVRP involves defining: (i) the amount of goods to be shipped from the depot to the satellites and from there to final customers, and (ii) the optimal routes connecting entities in each level, such that the capacity of the vehicles is not exceeded. Costs in two-echelon systems usually comes from two sources: traveling costs (which usually

depends on the distance traveled by the vehicles) and the cost of consolidating goods at the satellites. In many cases, 2E-CVRP aims at finding a collection of routes with minimum total cost (traveling plus consolidation costs).

In what follows we formalize the 2E-CVRP. To that end, let $G = (V, A)$ be digraph with set of vertices $V := \{\{0\}, S, C\}$, where 0 is a depot, $S := \{s_1, \ldots, s_k\}$ and $C := \{c_1, \ldots, c_n\}$ denote respectively sets of satellites and customers. Assume that, the arcs set A is partitioned into disjoint subsets A_1 and A_2, such that $A := A_1 \cup A_2$ ($A_1 \cap A_2 = \emptyset$). On the one hand, level-1 arcs, which connect the satellites and the depot, is given by set $A_1 := \{(i, j) : i, j \in \{0\} \cup S\}$. On the other hand, $A_2 := \{(i, j) : i, j \in C\} \cup \{(i, j) : i \in S, j \in C \text{ or } i \in C, j \in S\}$ denotes the set of level-2 arcs, which connect customers and satellites. A fleet $\mathcal{K}_1$ of $K_1 = |\mathcal{K}_1|$ homogeneous vehicles with capacity q_1 is in charge to implement level-1 routes, while level-2 routes are implemented by a fleet $\mathcal{K}_2$ of $K_2 = |\mathcal{K}_2|$ homogeneous vehicles with capacity Q_2. To each customer $i \in C$, an amount $d_i > 0$ of the same good must be delivered. Whenever an arc $(i, j) \in A$ is traversed by a vehicle, a cost c_{ij} is incurred. In addition, for each unit of good which passes on satellite s a cost L_s is going to be paid. The goal in 2E-CVRP is to assign up to K_1 level-1 routes and up to K_2 level-2 routes to ship goods from the depot the their respective customers. On the level-2 a given customer must be visited exactly once, while on the level-1 a given satellite may be visited n times ($n \geq 0$).

In Figure 2.3, we depict routes involved in a 2E-CVRP solution. In the figure, coloured circles represent elements of C, while colored squares denote elements of S. The picture indicates a collection of routes spanning 3 satellites, 12 customers and a single depot. Note that, there is no direct connection between the depot and customers. The transportation of goods follows the arcs in the digraph and only takes place between the depot and satellites and from there to the customers. Goods shipped from the depot must stop at satellites, where loads are consolidated into smaller vehicles. Accordingly, two types of routes are indicated. Those in the lower part of the picture, named level-1 routes, connect the depot to the satellites. The others, on the upper part, are named level-2 routes. They connect one single satellite to one or more customers.

As VRPs, various types of constraints associated with the routing and the consolidation of loads define some 2E-CVRP variants. In the literature, some authors interchangeably use the terms 2E-CVRP and 2E-VRP to refer to the basic version of problem, where only capacity constraints are considered. According to Perboli et al. [2008b], other variants include the 2E-VRP with Time Windows (2E-VRP-TW), where customers can be served within a given time windows interval. In the 2E-VRP with Satellites Synchronization (2E-VRP-SS) time constraints on the arrival and the depar-

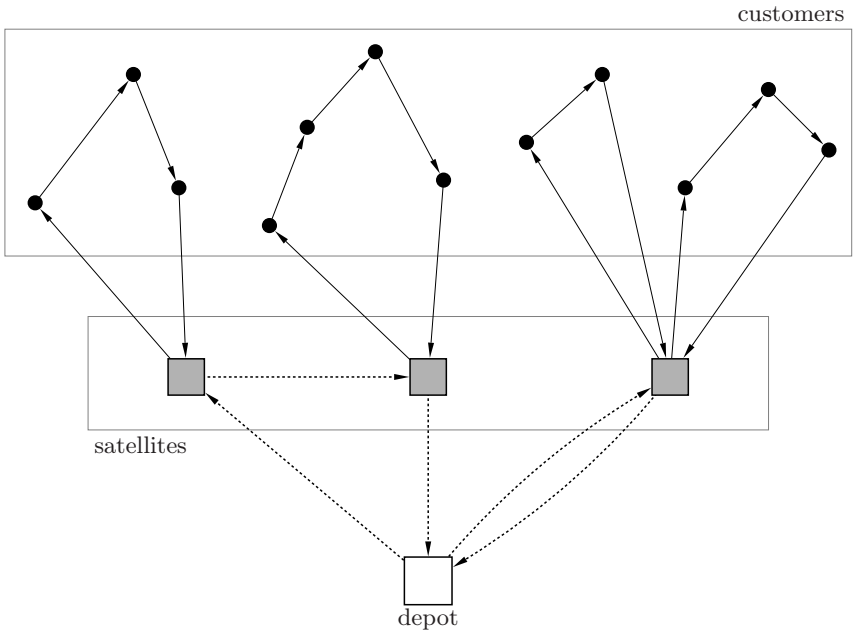


Figure 2.3: Example of solution for the 2E-VRP

ture of vehicles at the satellites are considered. In both problems, the 2E-VRP-TW and the 2E-VRP-SS, time windows constraints may be hard or soft. There are also the multi-depot 2E-VRP where loads can departure from more than one depot to satisfy customers demand. Finally, the 2E-VRP with Taxi Services (2E-VRP-TS) allows some loads to be shipped directly from the depot to customers, bypassing satellites.

From a computational point of view, solving 2E-CVRP is not a trivial task. That applies since the Capacitated Vehicle Routing Problem (CVRP), a widely known NP-hard optimization problem, can be seen as a 2E-CVRP special case, with only one depot and satellite. The first exact solution approach for the 2E-CVRP was introduced by Perboli et al. [2008b, 2011]. The authors proposed an Integer Programming (IP) formulation based on network flows. Such a formulation consists of defining flows from the depot to satellites and from there to customers. Variables f_{ij}^1 and f_{ij}^{2s} define the flow on the level-1 and level-2, respectively. Similarly, arc variables x_{ij}^1 represent the number of level-1 vehicles using arc $(i, j) \in A_1$ and binary variables x_{ij}^{2s} represent whether an arc is used or not for routing on level-2. Finally, variables z_i^s assume value 1 if and only if the demand of a given customer $i \in C$ pass through a satellite $s \in S$ in the solution. The model proposed by Perboli et al. [2008b, 2011] is described in the following. Aiming to lighten the presentation of such model, we represent the total amount of goods shipped from a satellite $s \in S$ as $D_s = \sum_{i \in C} d_i z_i^s$.

$$\sum_{(i,j)\in A_1} c_{ij}x^1_{ij} + \sum_{(i,j)\in A_2} c_{ij} \sum_{s\in S} x^{2s}_{ij} + \sum_{s\in S} L_s D_s \tag{2.61}$$

$$\sum_{s\in S} x_{0s} \le K_1 \tag{2.62}$$

$$\sum_{j\in 0\cup S} x^1_{jk} = \sum_{i\in 0\cup S} x^1_{ki} \qquad k \in 0\cup S \tag{2.63}$$

$$\sum_{s\in S}\sum_{i\in C} x^{2s}_{si} \le K_2 \tag{2.64}$$

$$\sum_{i\in C} x^{2s}_{si} \le m_s \qquad s \cup S \tag{2.65}$$

$$\sum_{i\in C} x^{2s}_{si} = \sum_{i\in C} x^{2s}_{is} \qquad s \cup S \tag{2.66}$$

$$\sum_{j\in 0\cup S} f^1_{ji} - \sum_{j\in 0\cup S} f^1_{ij} = \begin{cases} \sum_{i\in C} d_i, & i = 0 \\ D_i, & i \in S \end{cases} \qquad i \in 0\cup S \tag{2.67}$$

$$f^1_{ij} \le Q_1 x^1_{ij} \qquad (i,j) \in A_1 \tag{2.68}$$

$$\sum_{j\in S\cup C} f^{2s}_{ji} - \sum_{j\in S\cup C} f^{2s}_{ij} = \begin{cases} z^s_i, & i \in C \\ -D_i, & i \in S \end{cases} \qquad i \in S\cup C, s \in S \tag{2.69}$$

$$f^{2s}_{ij} \le Q_2 x^{2s}_{ij} \qquad (i,j) \in A_2, s \in S \tag{2.70}$$

$$\sum_{s\in S} f^1_{is} = 0 \tag{2.71}$$

$$\sum_{i\in C} f^{2s}_{is} = 0 \qquad s \in S \tag{2.72}$$

$$x^{2s}_{ij} \le z^s_i \qquad s \in S, i \in C, j \in S\cup C \tag{2.73}$$

$$x^{2s}_{ji} \le z^s_i \qquad s \in S, i \in S, j \in C \tag{2.74}$$

$$\sum_{i\in S\cup C} x^{2s}_{ij} = z^s_j \qquad s \in S, j \in C \tag{2.75}$$

$$\sum_{i\in S} x^{2s}_{ji} = z^s_j \qquad s \in S, j \in C \tag{2.76}$$

$$\sum_{s\in S} z^s_i = 1 \qquad i \in C \tag{2.77}$$

$$x^{2s}_{si} \le \sum_{j\in 0\cup S} x^1_{sj} \qquad s \in S, i \in C \tag{2.78}$$

$$x^2 \in \mathbb{B}^{|S||A_2|}, z \in \mathbb{B}^{|S||C|} \tag{2.79}$$

$$x^1 \in \mathbb{N}^{|A_1|} \tag{2.80}$$

$$f^1 \in \mathbb{R}^{|A_1|}_+, f^2 \in \mathbb{R}^{|S||A_2|}_+ \tag{2.81}$$

The objective function (2.61) is to minimize routing costs plus loading/unloading costs at satellites. Constraints (2.62)-(2.65) ensure the total number of vehicles used on each level does not exceed the limit of the fleet. The flow balance is guaranteed due to equalities (2.66), (2.67) and (2.69). Coupling constraints (2.68) and (2.70) are used to couple flow and arc variables. Constraints (2.71) and (2.72) are proposed to ensure that flows are not returned to satellites on level-1 and level-2, respectively. Constraints (2.73)-(2.77) controls variables z_i^s values, while constraints (2.78) couple arc variables on level-1 and level-2. Finally, constraints (2.79)-(2.81) defines decision variables space.

Besides of introducing the network flow model (2.61)-(2.81), Perboli et al. [2008b, 2011] also present two families of inequalities to strengthen it. One of them is the subtour breaking constraints. The other is based on network flow assignments. A Branch-and-cut algorithm that separates both sets of valid inequalities was also provided. Additional 2E-CVRP valid inequalities were proposed by Perboli and Tadei [2010]. Some of them were adapted from valid inequalities for CVRP. They were separated as cutting planes in a new 2E-CVRP Branch-and-cut implementation. The computational experiments conducted in Perboli and Tadei [2010] suggested, however, that their use did not improve by much the results for most of the 2E-CVRP instances. Overall, better results were obtained with the Branch-and-cut method in Perboli et al. [2008b, 2011], that does not make use of the new inequalities.

Another exact solution approach for 2E-CVRP is proposed by Jepsen et al. [2012]. They model the routing on level-1 as a Split Delivery CVRP (SDCVRP) and for the level-2 they use a slight modification of the Capacitated Location Routing Problem. Both problems are integrated in a single model. Because routing on level-1 does not match exactly as a SDCVRP, the model proposed by Jepsen et al. [2012] consists of a relaxed version of model (2.61)-(2.81). The lower bounds of the relaxed model keeps valid. However, upper bounds and feasible solutions do not. To deal with such problems the authors proposed a specific branch-and-cut algorithm, which embeds specific branching rules. In addition, a heuristic procedure is proposed to check whether a feasible solution for the relaxed model is valid for 2E-CVRP or not. Computational experiments conducted by the authors showed that results from such an approach dominate clearly those results provided by ?Perboli et al. [2011].

From the heuristic side, Perboli et al. [2008b, 2011] proposed a Mathematical Programming two-step heuristic that splits 2E-CVRP into a collection of CVRPs. In its first step, customer to satellite assignments provided by the LP relaxation of the network model are used to define which customers are supplied by each satellite. Once the assignments are available, the demands of goods that need to be shipped from the depot to each satellite are known. The second step thus consists in solving CVRPs:

one to route vehicles from the depot to the satellites (at the first layer) and others to route vehicles from the satellites to their assigned customers (at the second layer). Therefore, the heuristic consists in solving CVRPs in both levels by a combination of CVRP exact [Ralphs et al., 2002] and heuristic [Perboli et al., 2008a] procedures, and then combining the solution to each CVRP into a full 2E-CVRP solution.

Crainic et al. [2011] also implemented an heuristic that follows similar guidelines. It also consists in first defining the customer to satellite assignments and then solving a collection of CVRPs to construct an initial 2E-CVRP feasible solution. After that, a local search procedure is applied. Differently from the heuristic in Perboli et al. [2008b, 2011], in the construction phase, the assignment is defined by running an heuristic for clustering customers and satellites, according to their Euclidean distances. After the assignment is available, $|S| + 1$ CVRPs are solved. The local search then starts. It consists of swapping the assignment of customers to satellites and re-running the procedures for determining the solution to the CVRPs. This procedure is embedded in a meta-heuristic framework that provides diversification for the swapping strategies.

In a previous work [Santos et al., 2012a], we proposed a set partitioning IP formulation and two Branch-and-price algorithms for 2E-CVRP. The first algorithm prices routes that satisfy the elementarity condition, while the second method relaxes these constraints and prices q-routes [Christofides et al., 1981]. Both algorithms make use of a 2E-CVRP column generation heuristic that consists in the following. Instead of taking into account all possible columns (routes) in a set partitioning reformulation, a very few of them (namely those that are priced out at each Branch-and-price node) is taken into account. The heuristic therefore consists in solving, by a commercial Mixed Integer Programming package, a (restricted) set partitioning model, stated only in terms of this small set of columns. The heuristic is called at some nodes of the Branch-and-price enumeration tree. Since the restricted set partitioning model involves quite few columns and no pricing is implemented, the heuristic usually runs very fast.

Although new optimality certificates and new best upper bounds were provided in Santos et al. [2012a], many 2E-CVRP instances were left unsolved by both algorithms in that study (even after a ten thousand seconds time limit). For such instances, the algorithm that prices q-routes provided stronger lower bounds. That happened despite the fact that the lower bounds obtained when route elementarity is enforced are much stronger than those provided by the other reformulation. It turns out that pricing routes satisfying the elementarity condition is too time demanding. As a result, the Branch-and-price algorithm that prices q-routes was capable of evaluating many more Branch-and-bound nodes and ended up being capable of providing stronger lower bounds. On the primal side, much better results were obtained with the first algorithm.

That happened because many q-routes that did not satisfy the elementarity condition needed to be excluded from the restricted set partitioning model, before running the column generation heuristic.

Chapter 3

Branch-and-price algorithms for the Vehicle Routing Problem with Cross-Docking

Over the last decade and a half, Branch-and-price (BP) algorithms [Barnhart et al., 1998; Vanderbeck and Wolsey, 1996; Lübbecke and Desrosiers, 2005] have become an important tool to solve routing problems [Salani, 2005]. In this chapter, we describe two Integer Programming Formulations for the Vehicle Routing Problem with Cross-Docking (VRPCD) and due to their exponential number of variables, we implemented two BP algorithms to solve them. In the following, we define the problem in the section 3.1, then we discuss the first formulation and a detailed description of the algorithm implementation in the section 3.2. The second approach is discussed in the section 3.3, concerning modeling and implementation issues, with focus on the solution of the pricing problems. An extensive review on the algorithms' performance is given in the section 3.4 by means of computational results obtained by solving two sets of instances using the proposed algorithms. We conclude the chapter on section 3.5 where we also provide further research directions.

3.1 Problem definition

Let $G = (V, A)$ be a directed graph where $V = \{0\} \cup S \cup C$, $S = \{1, \ldots, n\}$ represents a set of suppliers, $C = \{1', \ldots, n'\}$ represents a set of of customers and 0 denotes the Cross-Docking (CD) warehouse. Assume that $A = A_S \cup A_C$ ($A_S \cap A_C = \emptyset$), where $A_S = \{(i, j) : i, j \in \{0, 1, \ldots, n\}\}$ denotes the set of arcs connecting the suppliers

as well as the depot and $A_C = \{(i,j) : i,j \in \{0,1',\ldots,n'\}\}$ denotes the set of arcs connecting customers and the depot. Let us assign costs $\{c_{ij} \geq 0 : (i,j) \in A\}$ to the arcs of G. Consider that $P = \{p_i := (i,i',q_i) : i = 1,\ldots,n\}$ denotes a set of transportation requests, each one representing a given (unsplittable) load $q_i \geq 0$ to be shipped from a supplier i to a customer i'. Consider as well that a set $\mathcal{K}$, containing K homogeneous vehicles (or trucks) of capacity Q, is given to guarantee that all requests will be satisfied (collected and delivered). Finally, assume that before delivering goods to customers, each vehicle must necessarily stop at the CD, in order to allow loads to change trucks and eventually facilitate their delivery. Whenever a load q_i moves from/to vehicle k to/from another vehicle, a cost $c_i^k \geq 0$ is incurred at the CD.

The VRPCD consists of finding $2K$ routes: K in charge of collecting loads and another K responsible to deliver loads. Each route in the first set of K routes must start at the depot and visit some suppliers, collecting the loads, without exceeding the vehicle's capacity. Accordingly, after some loads have changed trucks at the CD, each one of the other K routes must visit customers, to deliver the loads. The goal is then to find $2K$ routes that minimize the total cost (given by the sum of the costs of the arcs traversed by the vehicles and the cost of changing loads at the CD). Although in the definition of VRPCD there is a one-to-one correspondence between vertices in S and in C, the model can easily allow (by introducing appropriate artificial vertices) the case where several loads are supplied or consumed by the same location.

3.2 Branch-and-price Algorithm #1

In the VRPCD each vehicle performs two routes, one for collecting loads and another for delivering. Even though such routes depend from each other in the optimal solution, due to the loading/unloading costs, in this first solution approach we proposed to model such routes independently, because it allows us to decompose the problem into two integrated CVRP: one for visiting suppliers and another for visiting customers. The next mathematical formulation and BP algorithm explore such an structure.

3.2.1 Integer Programming Formulation

Let R (resp. R') denote the sets of all possible routes connecting the depot and all possible subsets of suppliers S (resp. of customers C). For each route $r \in R$ (resp. $r' \in R'$) let c_r (resp. $c_{r'}$) denote the sum of the costs of its arcs. Assume that we are given a binary parameter a_r^i (resp. $b_{r'}^{i'}$) that assumes value 1 whenever $i \in S$ is visited by r (resp. when $i' \in C$ is visited by r'). In order to formulate the VRPCD as an

Integer Program, let us use the following sets of decision variables: (a) λ_r^k (resp. $\gamma_{r'}^k$) to indicate whether or not vehicle $k \in \mathcal{K}$ performs route r to collect loads (resp. performs route r' to deliver loads) and (b) τ_i^k to indicate whether or not loads of request p_i moved from/to vehicle k before being delivered to its destination. An Integer Programming Formulation for VRPCD is (3.1)-(3.8):

$$\min \sum_{r \in R} c_r \sum_{k \in \mathcal{K}} \lambda_r^k + \sum_{r' \in R'} c_{r'} \sum_{k \in \mathcal{K}} \gamma_{r'}^k + \sum_{k \in \mathcal{K}} \sum_{i \in S} c_i^k \tau_i^k \tag{3.1}$$

$$\sum_{r \in R} \lambda_r^k = 1 \qquad \forall\, k \in \mathcal{K} \tag{3.2}$$

$$\sum_{r' \in R'} \gamma_{r'}^k = 1 \qquad \forall\, k \in \mathcal{K} \tag{3.3}$$

$$\sum_{r \in R} a_r^i \sum_{k \in \mathcal{K}} \lambda_r^k = 1 \qquad \forall i \in S \tag{3.4}$$

$$\sum_{r' \in R'} b_{r'}^{i'} \sum_{k \in \mathcal{K}} \gamma_{r'}^k = 1 \qquad \forall i' \in C \tag{3.5}$$

$$\sum_{r \in R} \lambda_r^k a_r^i - \sum_{r' \in R'} \gamma_{r'}^k b_{r'}^{i'} + \tau_i^k \geq 0 \qquad \forall p_i \in P,\ \forall\, k \in \mathcal{K} \tag{3.6}$$

$$-\sum_{r \in R} \lambda_r^k a_r^i + \sum_{r' \in R'} \gamma_{r'}^k b_{r'}^{i'} + \tau_i^k \geq 0 \qquad \forall p_i \in P,\ \forall\, k \in \mathcal{K} \tag{3.7}$$

$$\lambda \in \mathbb{B}^{K|R|}, \gamma \in \mathbb{B}^{K|R'|}, \tau \in \mathbb{B}^{Kn}. \tag{3.8}$$

The objective function (3.1) minimizes the total cost (routing plus changing loads at the CD). Convexity constraints (3.2) and (3.3) ensure that all K vehicles are used, while (3.4) and (3.5) enforce that each supplier and customer will be visited exactly once by a vehicle. Together, (3.6) and (3.7) force variable τ_i^k to assume value 1 whenever a request p_i is either collected and not delivered by vehicle k or when it is not collected but delivered by k. We emphasize that, such a modeling approach split the VRPCD into two disjoint CVRP, which are integrated by means of variables τ_i^k. On the one hand, dealing with separated problems lead us to symmetry problems on the formulation. On the other hand, such an approach may help us to reduce complexity on solving pricing problems, as described in next section.

3.2.2 Solving the Linear Programming relaxation by Column Generation

Let us consider the Linear Programming Master Program (LPMP), given by (3.1)-(3.7) and

$$\lambda_r^k \geq 0 \qquad \forall r \in R, \forall\, k \in \mathcal{K}, \tag{3.9}$$

$$\gamma_{r'}^k \geq 0 \qquad \forall r' \in R', \forall\, k \in \mathcal{K}, \tag{3.10}$$

$$0 \leq \tau_r^k \leq 0 \qquad \forall p_i \in P, \forall\, k \in \mathcal{K}. \tag{3.11}$$

Assume that initial subsets of routes $\hat{R} \subset R$ and $\hat{R}' \subset R'$ ($|\hat{R}| << |R|, |\hat{R}'| << |R'|$) replace respectively sets R and R' of the LPMP to formulate the Restricted Linear Programming Master Problem (RLPMP). Assume as well that a basic solution $\{\lambda_r^{k*} \in \hat{R}\}, \{\gamma_{r'}^{k*} \in \hat{R}'\}$ and $\{\tau_i^{k*} : \forall p_i \in P\}$ is available to the RLPMP and that $\{\alpha^k : k \in \mathcal{K}\}$, $\{\beta^k : k \in \mathcal{K}\}$, $\{\theta_i : i \in S\}$, $\{\mu_{i'} : i' \in C\}$, $\{\pi_i^k : k \in \mathcal{K}, p_i \in P\}$ and $\{\chi_i^k : k \in \mathcal{K}, p_i \in P\}$ denote dual variables respectively assigned to constraints (3.2)-(3.7). If the constraints

$$\alpha^k + \sum_{i \in S} a_r^i \theta_i + \sum_{i \in S} a_r^i \pi_i^k - \sum_{i \in S} a_r^i \chi_i^k \leq c_r \qquad \forall r \in R \setminus \hat{R}, \forall\, k \in \mathcal{K} \tag{3.12}$$

$$\beta^k + \sum_{i' \in C} b_{r'}^{i'} \mu_{i'} - \sum_{i' \in C} b_{r'}^{i'} \pi_i^k + \sum_{i' \in C} b_{r'}^{i'} \chi_i^k \leq c_{r'} \qquad \forall r' \in R' \setminus \hat{R}', \forall\, k \in \mathcal{K} \tag{3.13}$$

are satisfied, then $\{\lambda_r^{k*} : r \in \hat{R}\}, \{\lambda_r^{k*} = 0 : r \in R \setminus \hat{R}\}, \{\gamma_{r'}^{k*} : r' \in \hat{R}'\}, \{\gamma_{r'}^{k*} = 0 : r' \in R' \setminus \hat{R}'\}$ and $\{\tau_i^{k*} : \forall k \in \mathcal{K}, \forall p_i \in P\}$ solve the LPMP. Otherwise, sets $\hat{R}$ and $\hat{R}'$ must be enlarged with those routes that respectively violate (3.12) and (3.13). A new RLPMP is formulated and re-optimized. The process is repeated until constraints (3.12) and (3.13) are satisfied.

3.2.3 Column Generation Subproblem

We call the problem to identify violated dual constraints (3.12) and (3.13) (resp. negative reduced cost variables λ_r^k and $\gamma_{r'}^k$ on the RLPMP) as column generation subproblem or pricing problem. We price out negative reduced cost variables λ_r^k by adding the dual variables values $-\theta_j - \pi_j^k + \chi_j^k$ on each arc $(i,j) \in A : j \in S$ for each vehicle and solving a Elementary Shortest Path Problem with Resource Constraints (ESPPRC)

[Feillet et al., 2004] on the resulting graph. Variables $\gamma_{r'}^k$ are priced out in a similar fashion using the dual variables values $-\mu_j + \pi_j^k - \chi_j^k$ on arcs of G.

In what follows, we describe a Dynamic Programming (DP) algorithm for solving the ESPPRC for the suppliers and the CD in order to price out λ_r^k variables. The algorithm is also applied to the case where only the customers and the CD are considered, for pricing $\gamma_{r'}^k$ variables.

Starting from an initial path that includes only the CD, the algorithm builds new paths from existing ones. A label L is assigned to each path kept in a list of paths. The label stores the load $q(L)$ of the path (the sum of the loads in the vehicle, at an intermediate or at the end vertex), its end node $e(L)$ (the last supplier visited by the path), a set $V(L) \subseteq S$ of suppliers visited by the path and the accumulated reduced cost $c(L)$ along the path, given by the sum of the traversed arcs of the path.

Like in most DP algorithms, dominance rules are used to identify states (paths) that, being provably suboptimal, do not need to be extended. To be more specific, assume that L_1 and L_2 are two labels assigned to two paths. We say that L_1 dominates L_2 if both terminate at the same vertex, the reduced cost of L_1 does not exceed the reduced cost of L_2 and if all feasible extensions to L_2 are also feasible to L_1. As an example, suppose $e(L_1) = e(L_2)$. Label L_1 dominates L_2 if:

$$\begin{array}{c} c(L_1) \leq c(L_2), \\ q(L_1) \leq q(L_2), \\ V(L_1) \subseteq V(L_2) \end{array} .$$

A path of label L can be extended to $j \in S$ if: $(e(L), j) \in A$, $q(L) + q_j \leq Q$.

3.2.4 Algorithm's Implementation Details

As described above, those routes violating constraints (3.12) or (3.13) are returned by the pricing algorithm and appended to sets $\hat{R}$ or $\hat{R}'$, respectively. So, a new RLPMP is formulated and reoptimized. Although our pricing algorithm is capable of returning all violated routes on each execution, we do not add all of them to the RLPMP. Instead, we select the 10% best routes according to the reduced cost to include on the RLPMP. Such an strategy aims to accelerate the convergence of column generation algorithm, without making the RLPMP dense and harder to solve. After a given route is appended to sets $\hat{R}$ or $\hat{R}'$ it remains there until the algorithm stops.

After evaluating the LPMP (the LP bound of model (3.1)-(3.8)) using the column generation approach, if the solution in hands is integer, the problem is solved. Otherwise, we must resort to branching. In our implementation, we branch on variables τ_i^k

because they have a stronger impact on RLPMP. In fact, variables τ_i^k integrate those disjoint CVRP for suppliers and customers and for this reason, by fixing such variables we also modify the routing structure. Assuming that sets $\hat{R}$ and $\hat{R}'$ were used to evaluate the last RLPMP, the variable τ_i^k chosen to branching must be that on which indexes p_i and k maximize the amount $\sum_{r\in R} \min\{a_r^i\lambda_r^k, 1-a_r^i\lambda_r^k\}+\sum_{r'\in R'} \min\{b_{r'}^{i'}\gamma_{r'}^k, 1-b_{r'}^{i'}\gamma_{r'}^k\}$. Such a value measure the uncertainty of vehicle k to attend request p_i. However, we recall that by fixing $\tau_i^k = 0$, we only ensure that p_i does not change from/to vehicle k at the CD. Such an equality by itself cannot define precisely if k collects and delivers p_i or if it neither collects nor delivers p_i. The same holds by fixing $\tau_i^k = 1$, because such a constraint is not able define whether vehicle k collects p_i without delivering it or the opposite.

Thus, four nodes need to be created to address properly our branching strategy. They are described in the following table:

node	associated constraints
i	$\tau_i^k = 0,\ \sum_{r\in R} a_r^i\lambda_r^k = 1,\ \sum_{r'\in R'} b_{r'}^{i'}\gamma_{r'}^k = 1$
ii	$\tau_i^k = 0,\ \sum_{r\in R} a_r^i\lambda_r^k = 0,\ \sum_{r'\in R'} b_{r'}^{i'}\gamma_{r'}^k = 0$
iii	$\tau_i^k = 1,\ \sum_{r\in R} a_r^i\lambda_r^k = 1,\ \sum_{r'\in R'} b_{r'}^{i'}\gamma_{r'}^k = 0$
iv	$\tau_i^k = 1,\ \sum_{r\in R} a_r^i\lambda_r^k = 0,\ \sum_{r'\in R'} b_{r'}^{i'}\gamma_{r'}^k = 1$

Node i indicates that vehicle k collects and delivers request p_i; node ii indicates that vehicle k neither collects nor delivers request p_i; node iii indicates that vehicle k collects but does not deliver request p_i; finally, node iv indicates that vehicle k delivers request p_i but does not collect it.

Proposition: **Constraints (3.2)-(3.7) together with branching constraints i-iv ensure integral solutions for the VRPCD**

Suppose a leaf node on the BP tree including the above branching constraints for all vehicles and requests, leading to a set of feasible assignments $\tau_i^k = \{0, 1\}$, $\forall p_i \in P$, $k \in \mathcal{K}$. With such assignments and the associated branching constraints is easy to see that the following equality holds for the suppliers

$$\sum_{r\in\hat{R}} a_r^i\lambda_r^k = \{0,1\} \quad \forall i \in S,\ k \in \mathcal{K}. \tag{3.14}$$

Let us take the solution for a given vehicle k^*. On the one hand, constraints (3.2) ensure that precisely one route (visiting at least one supplier) must be associated with k^* to collect requests. On the other hand constraints (3.4) ensure that each supplier

i must be visited for exactly one vehicle. Thus, constraints (3.2), (3.4) and (3.14) together state that

$$\exists i \in S : \sum_{r \in \hat{R}} a_r^i \lambda_r^{k^*} = 1 \text{ and } \sum_{k \in \mathcal{K} \setminus k^*} \sum_{r \in \hat{R}} a_r^i \lambda_r^k = 0. \tag{3.15}$$

Let us now consider a non-empty set $I^* \subseteq S$ composed by those suppliers visited by vehicle k^*. According to (3.15) no other vehicles rather than k^* visit elements of I^*. Because $\hat{R}$ comprises only elementary routes and due to the triangle inequality property on the arcs costs of G we have that: only one route spanning all vertices of I^* is assigned to vehicle k^* or a set of routes spanning all vertices of I^* is associated with vehicle k^* (all routes have the same cost due to the objective function (3.1)). In the last case we can transform the fractional solution into integral by selecting only one route of the set and assigning to its respective variable λ a integral value.

The above steps can be generalized to prove integrality of solutions for every vehicle as well as for customer vertices. Whether we have variables λ and γ assuming integral values, variables τ are also integral due to constraints (3.6) and (3.7). Therefore, the proposed branching rule ensures integral solutions for the BP algorithm. □

In order to define the initial sets $\hat{R}$ and $\hat{R}'$ (to formulate the first RLPMP), we generate K feasible routes to collect the loads and another K feasible routes to deliver them. Such routes are generated randomly choosing sets of suppliers (customers) whose sum of loads does not exceed Q.

At the end of the root node, we implemented a Column Generation Heuristic (CGH), which consists of replacing sets R and R' of model (3.1)-(3.8) by the sets of routes available after the LPMP to be solved. A corresponding Restricted Integer Program is formulated and solved by a LP based Branch-and-bound algorithm. The search of nodes along the branch-and-price tree is conducted according to the best-first strategy.

3.3 Branch-and-price Algorithm #2

The BP algorithm described in the previous section decomposes the VRPCD into two integrated CVRP. Although this approach allows one to solve smaller subproblems, the model (3.1)-(3.8) is plagued with a deep symmetry, because routes are indexed by vehicles. Such a shortcoming complicates the algorithm's convergence, especially when the number of vehicles increases.

To deal with such a shortcoming we propose an alternative Integer Programming Formulation, avoiding to index routes by vehicles. Indeed, in this second approach we consider that a given vehicle performs a complete route, which includes the collect of loads at suppliers, loading/unloading operations at the CD as well as the delivering of loads to their respective customers. We also propose a BP algorithm to solve the formulation, which is described along this section.

3.3.1 Integer Programming Formulation

Let R be the set of all feasible routes in G leaving the CD for visiting a set of suppliers, returning to the CD for consolidation of loads and then delivering loads to their respective customers. For each route $r \in R$ let c_r denote its cost, given by the sum of its arcs. Assume we are given parameters a_{ir} indicating if route $r \in R$ visits or not vertex $i \in V$ (with value 1 or 0, respectively) and $b_{ir} = \max(0, a_{i'r} - a_{ir})$ that assumes value 1 if and only if route r visits customer i' and do not visit the respective supplier i of a given request $p_i \in P$. The column generation formulation is accomplished by using decision variables λ_r indicating if route r is used or not in the solution, with value 1 or 0, respectively, and decision variables τ_i to control the changing of loads at the CD, assuming value 1 if the customer i' of request p_i lies on the route r but the respective supplier i does not, and 0 otherwise. An alternative Integer Programming Formulation for the VRPCD follows.

$$\min \quad \sum_{r \in R} c_r \lambda_r + \sum_{p_i \in P} c_i \tau_i \tag{3.16}$$

$$\sum_{r \in R} \lambda_r = K \tag{3.17}$$

$$\sum_{r \in R} a_{ir} \lambda_r = 1 \qquad \forall i \in V \setminus \{0\} \tag{3.18}$$

$$\tau_i - \sum_{r \in R} b_{ir} \lambda_r \geq 0 \qquad \forall p_i \in P \tag{3.19}$$

$$\lambda \in \mathbb{B}^{|R|}, \tau \in \mathbb{B}^n. \tag{3.20}$$

The objective function (3.16) minimizes the sum of routing costs plus loading/unloading costs at the CD. Convexity constraint (3.17) assures that all K vehicles are used in the routing solution, while equalities (3.18) impose that routes on set R must be selected for spanning all suppliers and customers of G. Inequalities (3.19) control loading/unloading operations at the CD, enforcing $\tau_i = 1$ if and only if some route

visits a customer without visiting the respective supplier. Finally, constraints (3.20) define the decision variables bounds. Noteworthy is the fact that the loading/unloading costs c_i on the above formulation differ from that proposed in the model (3.1)-(3.8). Here such costs are incurred whenever a vehicle delivers the request p_i without been collected it, while in the previous model the cost c_i^k is incurred if the load of request p_i is loaded/unloaded from/to a vehicle at the CD. Indeed, if we define $c_i = 2c_i^k$ both approaches become equivalent.

3.3.2 Solving the Linear Programming relaxation by Column Generation

Let us now formulate a LPMP for the second model, which is given by (3.16)-(3.19) and

$$\lambda_r \geq 0 \qquad \forall r \in R \tag{3.21}$$

$$0 \leq \tau_r^k \leq 0 \qquad \forall p_i \in P. \tag{3.22}$$

The respective RLPMP is obtained by considering a restricted set of routes $\hat{R} \subset R$ on the LPMP instead of set R. If we associate dual variables $\alpha \in \mathbb{R}$, $\{\theta_i \in \mathbb{R} : i \in \{1, \ldots, n, 1', \ldots, n'\}\}$ and $\{\chi_i \in \mathbb{R}_+ : i = 1, \ldots, n\}$ respectively to constraints (3.17), (3.18) and (3.19), we obtain a dual formulation of RLPMP given by (3.23)-(3.25):

$$\max \qquad K\alpha + \sum_{i \in V \setminus \{0\}} \theta_i \tag{3.23}$$

$$\alpha + \sum_{i \in V \setminus \{0\}} a_{ir}\theta_i - \sum_{p_i \in P} b_{ir}\chi_i \leq c_r \qquad \forall r \in \hat{R} \tag{3.24}$$

$$\chi_i \leq c_i \qquad \forall p_i \in P. \tag{3.25}$$

Suppose $\{\lambda_r^* : r \in \hat{R}\}$ optimizes the RLPMP. We must look for reduced cost routes $r \in R \setminus \hat{R}$ violating constraints (3.24). If such routes exist, they are added in the set $\hat{R}$ and the RLPMP is re-optimized. Otherwise, we evaluate the LPMP with variables $\{\lambda_r^* : r \in \hat{R}\}$, $\{\lambda_r^* : r \in R \setminus \hat{R}\}$ and the column generation stops.

3.3.3 Column Generation Subproblem

Before to discuss how we formulate and solve the column generation subproblem, let us first introduce a small modification performed on the set of arcs in order to facilitate the implementations. The problem is formulated on graph $G' = (V, A')$ where

$$A' = (\, A \setminus (A_{S0} \cup A_{0C})\,) \cup A_{SC}$$
$$A_{S0} = \{(i,0)\,\forall i \in S\}, \quad A_{0C} = \{(0,j)\,\forall j \in C\}, \quad A_{SC} = \{(i,j) : i \in S, j \in C\}.$$

We also assign costs $c_{ij} = c_{i0} + c_{0j},\ \forall (i,j) \in A_{SC}$. Note that, arcs from suppliers to the CD and from the CD to customers in A are replaced by arcs connecting directly suppliers to customers in A'. For this reason, routes in G' must leave the CD to collect products at suppliers and visit customers immediately latter, returning to the CD at end of route. We model loading/unloading operations at the CD by adding a dual cost χ_i whether a given vehicle (route) visits a customer i' without visiting the respective supplier i of request $p_i \in P$. We illustrate such modifications in the Figure 3.1, where we depict a route collecting requests p_i and p_j at suppliers and delivering the requests p_i and p_z to their respective customers. In this example, only the cost χ_z is incurred in the route, because it does not collect p_z, but does deliver. Another route is going to pay the cost χ_j incurred to deliver p_j without collecting it.

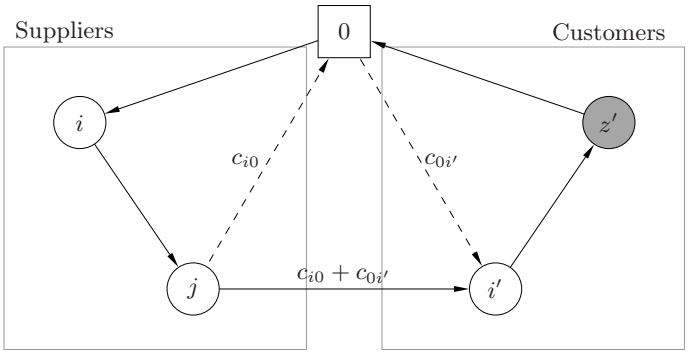


Figure 3.1: Modifications on the graph to solve the pricing problem

The associated column generation subproblem is a slightly variant of the ESPPRC. Such a variant occurs due to the loading/unloading costs incurred when a route (elementary path) visits a customer vertex i' without visiting the respective supplier i. Therefore, the optimal path must be evaluated looking for routing and loading/unloading costs, which seems to increase the problem complexity.

We propose different algorithms for solving the pricing problem. The first of them consists of modifying the dominance rules of the DP algorithm for the ESPPRC described in the section 3.2.3 to cope with the loading/unloading costs along the path

toward to optimal solutions. In addition to the new dominance rules, we also implemented a different strategy to investigate labels. Such a strategy is a bidirectional search proposed by Salani [2005], which consists of creating labels from source to destination and from destination to source together. The algorithm implementation is described in section 3.3.3.1. According to Jepsen et al. [2008], Branch-and-Cut (BC) algorithms tend to outperform DP algorithms when the dominance rules are loose, since a large amount of labels tend to be created and extended, slowing their convergence. Such a shortcoming can be realized when there are few resources or they are not well restricted, for example. We agree with Jepsen et al. and also implemented a BC algorithm for pricing routes, to be introduced in section 3.3.3.2. Although BC algorithm can perform better than the DP algorithm whether resource constraints are not tight, it prices fewer routes for iteration, slowing the BP algorithm convergence. To solve this problem we also describe in section 3.3.3.3 the implementation of a heuristic algorithm based on GRASP (Greedy Randomized Adaptive Search Procedure) [Feo and Resende, 1995] meta-heuristic to solve the subproblem. Using the heuristic and the BC algorithms together we can price out a large number of routes faster, helping the BP algorithm to converge faster for some loose constrained instances.

In what follows, we modify graph G' by adding an artificial copy of the depot $0'$, in order to look for paths instead of routes. We also embed the optimal values of dual variables θ on arcs costs of A' obtaining $c'_{ij} = c_{ij} - \theta_j, \forall (i,j) \in A'$ to facilitate the algorithms' implementations.

3.3.3.1 Dynamic Programming Algorithm

Let us now discuss a DP algorithm for pricing routes of the new formulation, which concerns suppliers and customers in the same route (including loading/unloading costs). Due to the modifications on the graph, such an algorithm includes new dominance rules and implements a bidirectional search strategy [Salani, 2005] to investigate labels aiming to evaluate solutions faster. The definition of labels here is the same from that introduced in the previous DP algorithm. However, two kinds of labels are needed to implement the bidirectional search: forward and backward labels.

Forward labels (L^F) represent paths which start at the source node and visit a subset of suppliers $\overline{S} \subseteq S$ toward the destination. Likewise, backward labels (L^B) are used to store paths that start at destination node and visit a subset of customers $\overline{C} \subseteq C$, towards the source node. In our implementation, forward labels are forbidden to visit customer nodes. Similarly, backward labels are not allowed to be extended to any supplier. Our algorithm starts determining all feasible labels L^F and then L^B.

Differently from other implementations of DP bidirectional search, the order in which backward or forward labels are determined is not relevant, since they are respectively forbidden to visit suppliers and customers. Feasible labels L^F and L^B are generated by extending paths, according to the same rules used by the previous DP algorithm. After L^F and L^B labels are determined (with the application of dominance rules to be described shortly), we attempt to merge pairs of forward and backward labels, in order to build a full feasible path from source to destination.

The dominance rules for pruning labels L^F work as follows (dominance rules for pruning labels L^B are symmetrically defined). Assume, as before, that L_1^F and L_2^F are two labels assigned to the paths being compared, $\overline{S}_1, \overline{S}_2$ are the suppliers visited by them and $\overline{A}_1, \overline{A}_2 \subseteq A$ denote the subsets of arcs traversed by the paths. Recall that both L_1^F and L_2^F only visit vertices of S. From now on, assume that both paths end in vertex $j \in S$, i.e., $v(L_1^F) = v(L_2^F) = j \in S$. For guaranteeing that L_1^F dominates L_2^F, we must have: (i) $\overline{S}_1 \subseteq \overline{S}_2$ (which implies that $q(L_1^F) \leq q(L_2^F)$) and (ii) $\sum_{(i,j)\in\overline{A}_1} c'_{ij} + \sum_{i\in\overline{S}_2\setminus\overline{S}_1} \chi_i^* \leq \sum_{(i,j)\in\overline{A}_2} c'_{ij}$. Note that in the latter expression, a very conservative assumption was made: we assume that there is a non-dominated backward label L^B, responsible to deliver all loads $\{q_i : i \in \overline{S}_2 \setminus \overline{S}_1\}$, which are collected by L_2^F and not collected by L_1^F. In this case, L_1^F dominates L_2^F if the cost (including the eventual cost of loading/unloading operations) of L_1^F is lower than that of L_2^F.

As stated before, the performance of DP algorithms depends on how restricted are the problems. On the one hand, since $\chi_i^* \geq 0 : i = 1, \ldots, n$, when the term $\sum_{i\in\overline{S}_2\setminus\overline{S}_1} \chi_i^*$ is included in the left hand side of condition (ii) above, the dominance conditions are weakened. Thus, the algorithm tends to perform poorly whenever loading/unloading costs increase. We recall that, whether $c_i = 0 : i = 1, \ldots, n$, the above subproblem becomes the classical ESPPRC, since only the routing costs are considered. On the other hand, since the capacity constraints is the single resource to restrict the label extensions, whether such a resource is not tight, much more labels are created and extended, leading to the algorithm performs slower. We conclude that, DP algorithm is highly sensitive to the instances properties. For this reason, we implemented a BC algorithm as an alternative approach for solving the pricing problem, which is described next.

3.3.3.2 Branch-and-cut Algorithm

Our BC algorithm lies on an arc formulation which uses three types of decision variables. One type concerns binary decision variables x_{ij} to control whether an arc $(i, j) \in A'$ is selected or not in the solution, assuming respectively value 1 or 0. The

other consists of decision variables y_i to indicate whether a vertex $i \in V$ is visited (or not), with value 1 (or 0). Finally, we also make use of binary decision variables t_i which assume value 1 if, for a given request p_i, the customer i' is visited and its respective supplier i is not, 0 otherwise. The Integer Programming Formulation is given by (3.26)-(3.35):

$$\min \sum_{(i,j)\in A} c'_{ij}x_{ij} \quad + \quad \sum_{p_i\in P} c_i t_i \tag{3.26}$$

$$\sum_{j\in S} x_{0j} = 1 \tag{3.27}$$

$$\sum_{i\in C} x_{i0'} = 1 \tag{3.28}$$

$$\sum_{j\in\Gamma_i^+} x_{ij} = -y_i \qquad \forall\, i \in V \setminus \{0, 0'\} \tag{3.29}$$

$$\sum_{h\in\Gamma_i^-} x_{hi} = -y_i \qquad \forall\, i \in V \setminus \{0, 0'\} \tag{3.30}$$

$$\sum_{i\in W,\, j\in V-W} x_{ij} \geq y_j \qquad \forall\, 0 \in W \subset V,\, \forall j \in V \setminus W \tag{3.31}$$

$$t_i \geq y_i - y_{i'} \qquad \forall p_i \in P \tag{3.32}$$

$$\sum_{i\in S} y_i q_i \leq Q \tag{3.33}$$

$$\sum_{i'\in C} y_{i'} q_{i'} \leq Q \tag{3.34}$$

$$x \in \mathbb{B}^{|A'|}, y \in \mathbb{B}^{|V|}, t \in \mathbb{B}^n \tag{3.35}$$

The objective function (3.26) minimizes the sum of arcs costs plus loading/unloading costs. Equalities (3.27) and (3.28) ensure that exactly one arc is used to leave and arrive the depot and its artificial copy, respectively. From (3.29) and (3.30) we have the couple constraints to assure together that if any arc of a given vertex is used such a vertex must be visited. Directed cutset constraints (3.31) enforce connectivity and subtour elimination on solutions. The constraints (3.32) control variables t_i associated with loading/unloading costs, while (3.33) and (3.34) are the capacity constraints used respectively to prevent a vehicle from collecting (or delivering) more loads than its capacity. Decision variables space is given by constraints (3.35).

The above model holds an exponential number of constraints due to (3.31). Our BC algorithm concerns solving model (3.26)-(3.35) disregarding constraints (3.31) and adding violated constraints by demand, until no such constraints are found. For BC

implementation we use callbacks routines of `CPLEX` optimization software (release 12.1) keeping all default parameters, except pre-processing and heuristic routines to prevent improper feasible solutions, since we deal only with a subset of constraints (3.31).

To identify violated constraints, we use `CPLEX` cut callback routine to take values of decision variables y_i^* and x_{ij}^* and build a graph $G_o = (V_o, A_o)$, where

$$V_o = \{0\} \cup \{i : y_i^* > 0\} \text{ and } A_o = \{(i,j) : x_{ij}^* > 0\}.$$

Then, we solve $|V_o| - 1$ max-flow (min-cut) problems on G_o from vertex $s = 0$ to $t_i = \{i \in V_o \setminus \{0\}\}$, assigning arcs capacities $C_{ij} = x_{ij}^*$, $\forall\, (i,j) \in A_o$.

Let $f(0, t_i)$ be the max-flow from vertex 0 to a given t_i on G_o. Violated constraints (3.31) are identified by looking for solutions on which $f(0, t_i) < y_i^* : i \in V_o$. On such a solution, let f_{ij} denote the flow passing through arc $(i,j) \in A_o$. To identify those arcs to compose a violated constraint, we build a residual graph $G_r = (V_o, A_r)$, where set $A_r = \{(i,j) \in A_o : C_{ij} - f_{ij} > 0\}$ comprises arcs with residual capacity of flow. Every path from 0 to t_i in the residual graph can be mapped on a set W, and thus, violated constraints (3.31) may be generated.

According to the definition of Directed cutset constraints, for each set W identified using the above procedure, we could generate cuts considering all vertices $i \in V \setminus W$. However, in our implementation we append on the model only the most violated cut for a given set W. Such a procedure is repeated until no more violated cuts can be found.

3.3.3.3 Heuristic Algorithm

GRASP meta-heuristic [Feo and Resende, 1995] consists basically of embedding random characteristics within greedy search procedures, aiming to achieve good quality local minima. It concerns a construction phase followed by a local search step. In the construction phase feasible solutions are built using a Restricted Candidate List (RCL) and a parameter $0 \leq \alpha \leq 1$ to control the greediness to choose its elements. After that, local search procedures are applied to that solution in order to achieve a local minimum. This procedure is repeated until a stopping criterion is achieved and the best solutions are stored along the iterations.

In what follows, we introduce an important notation used along this section. The subset $\Gamma_i^+ \subset V$ denotes a set of feasible vertices for which $i \in V$ can visit (feasible adjacent vertices of i) in a given iteration. We say that a vertex j belongs on set Γ_i^+ if, and only if, $(i,j) \in A'$, j was not visited by the incumbent path and the load q_j does not exceed the residual capacity of vehicle. Similarly, set $\Gamma_i^- \subset V$ comprises those

vertices able to visit i in a given iteration.

The algorithm starts the construction phase including vertex 0 on the path and filling up RCL with the best adjacent vertices, according to the parameter α. An adjacent is selected to be included on the path and another iteration takes place, until vertex $0'$ is selected, ending the construction phase. The algorithm fulfills the RCL based on the estimative costs of the adjacent vertices. Suppose i is the last vertex of a given partial path. The algorithm evaluates estimative costs a_j, $\forall j \in \Gamma_i^+$ such as

$$a_j = \begin{cases} c_{ij} + \min_{k \in \Gamma_j^+}(c_{jk}); & \text{if } j \in S \\ c_{ij} + \min_{k \in \Gamma_j^+}(c_{jk}) + b_j c_j; & \text{if } j \in C \\ c_{i0'}; & \text{if } j = 0' \end{cases}$$

The parameter b_j in the above expression has a similar meaning of that introduced in the section 3.3.1. It indicates whether the partial route delivers the request p_j without been collected it or not, assuming respectively value 1 or 0. The algorithm then calculates minimum $\lfloor a_j \rfloor = \min_{j \in \Gamma_i^+}(a_j)$ and maximum $\lceil a_j \rceil = \max_{j \in \Gamma_i^+}(a_j)$ estimative costs and fill up the RLC with those adjacents of i in the interval $\lfloor a_j \rfloor \leq a_j \leq \lfloor a_j \rfloor + \alpha \times (\lceil a_j \rceil - \lfloor a_j \rfloor)$. Note that, assigning $\alpha = 0$ we perform a greedy construction phase, while $\alpha = 1$ makes the search totally random.

For each solution built on the construction phase we apply local search movements in order to achieve its local minimum. The local search we implemented is a Variable Neighborhood Decent (VND) [Mladenovic and Hansen, 1997]. Instead of using a single neighborhood, VND uses H not necessarily related neighborhoods, applying to a given solution a larger number of movements. Suppose $f(x)$ is a cost function for a given solution x and $N_1, N_2, \ldots, N_H$ are neighborhoods. The next algorithm describes the VND local search.

Algorithm 1 VND Local Search

Require: Initial solution x_0
Ensure: Local minimum solution x_o^*
  $x \leftarrow x_0$; $h \leftarrow 1$;
  while $h \leq H$ **do**
    if ($\exists\, s \in N_h(x) : f(s) < f(x)$) **then**
      $x \leftarrow s$; $h \leftarrow 1$; **break;**
    end if
    $h \leftarrow h + 1$
  end while
  return x

In our heuristic we define $H = 3$ neighborhoods to be used within VND, which

are described as N_1, N_2 and N_3 in the following. Let V_r be a set of vertices visited by a given route r. N_1 is an insertion neighborhood composed by routes $\tilde{r}$ on which $V_{\tilde{r}} = V_r \cup \{i\} : i \in V \setminus V_r$. Solutions $\tilde{r} \in N_2$ consists in removing one vertex of solution r for which $V_{\tilde{r}} = V_r \setminus \{i\} : i \in V_r$ and finally N_3 is a swap neighborhood where vertices visiting order of a route r is changed. The movements are performed in the respective neighborhood only if they are feasible using the best improving strategy.

To help intensification of solutions obtained with GRASP heuristic we implemented a path-relinking algorithm [Glover, 1997]. Path-relinking uses solutions from a set of good quality solutions, named elite set, to guide the search for other solutions along solutions space. For our GRASP implementation, path-relinking consists in using an incumbent solution i obtained from each iteration of GRASP (construction phase + VND local search) and applying movements according to the neighborhoods N_1, N_2 and N_3 in order to transform it into a solution e, chosen from elite set. Different strategies are available to build the set of elite solutions, as well as to choose a solution from this set as guiding, but in our implementation we store in the elite set only the overall minimum cost solution. Suppose $I \subset R$ is the set of intermediate solutions obtained by converting i into e using path-relinking movements. If $\exists\, i^* \in I : f(i^*) < f(i)$, we apply VND local search on i^* to ensure it achieves its local minimum.

3.3.4 Algorithm's Implementation Details

The BP algorithm starts evaluating the LP bound of model (3.16)-(3.20) by solving the LPMP using the column generation approach described in sections 3.3.2 and 3.3.3. For this pursuit, we start the RLPMP with K randomly generated routes on set $\hat{R}$, one for each vehicle. After that, we price out negative reduced cost routes using or the DP algorithm or GRASP + BC algorithms. For the last option, GRASP heuristic is always called before BC. If GRASP fails in finding negative reduced cost routes, then BC is called. Independently of the pricing algorithm, whether negative reduced cost routes are found, they are appended to set $\hat{R}$ and a new iteration takes place. Otherwise, if neither DP nor BC find such routes, the LP bound is evaluated.

As in the BP#1, here again we evaluate a subestimative lower bound to be used when the LP bound of model (3.16)-(3.20) could not be evaluated in a reasonable time. Supposing w_{CG} as the optimal value evaluated by solving the RLPMP for a given set $\hat{R}$. The subestimative is evaluated as $w^*_{CG} = w_{CG} + K\rho(r^*)$, where $\rho(r^*)$ is the reduced cost of the optimal route $r^* \in R \setminus \hat{R}$.

After evaluating the LPMP, if all variables λ present integer values (variables τ are also integer due to constraints (3.19)), the optimal solution for the VRPCD is

achieved and BP algorithm stops. Otherwise, we must resort to branching. In this BP algorithm, we branch on arc variables x_{ij} obtained from the RLPMP as

$$x_{ij} = \sum_{r \in \hat{R}} a_{ijr} \lambda_r$$

where the parameter a_{ijr} indicates whether a route r pass through the arc $(i,j) \in A'$ or not, assuming respectively value 1 or 0. We branch on the most fractional arc $(i,j) \in \arg\min_{(i,j)\in A}\{\left|x_{ij} - \frac{1}{2}\right|\}$. In case of ties, we select the first arc in lexicographic order. After that, we create two nodes on the BP tree: one by fixing $x_{ij} = 1$ and another by imposing $x_{ij} = 0$. However, we rewrite these constraints in terms of λ values to keep the structure of the pricing problems. Thus, we include the constraints $\sum_{r\in\hat{R}} a_{ijr}\lambda_r = 1$ and $\sum_{r\in\hat{R}} a_{ijr}\lambda_r = 0$ in the RLPMP. For node selection on BP tree, we use the best bound strategy.

Aiming to improve upper bound values, once again we implemented a CGH. Accordingly, the set R on model (3.16)-(3.20) is replaced by set $\hat{R}$ after the LPMP is solved. The resulting Integer Programming Problem is solved using the MIP solver of `CPLEX` and an upper bound for the VRPCD is evaluated.

3.4 Computational Experiments

In this section, we report our computational experience with the BP algorithms described along this chapter. The algorithms were tested using instances generated from real test cases, coming from a Danish consulting firm, described in Wen et al. [2009]. Such test cases comprise five instances (indexed by a, b, c, d, e) containing 200 customers and suppliers (400 vertices). Because instances of this size are out of reach for current exact solution approaches, we extracted $n \in \{10, 15, 20, 25, 30\}$ suppliers and customers, to generate a new instance set for VRPCD. Arc costs represent the two dimensional Euclidean distance between their endpoints. For each instance, we solved a Bin Packing heuristic looking for the lowest number of vehicles to meet all requests. For each value of n we present the associated number of vehicles K (within brackets): $10(4), 15(6), 20(7), 25(9), 30(10)$.

Because our problem depends on the loading/unloading costs at CD, we consider here three different values: $c_i = 0, 20, 40$. For a given instance, the same loading/unloading costs were imposed to all goods. Such costs were defined empirically, in order to allow goods to be loaded/unloaded during the consolidation step at CD. On the one hand, setting values of $c_i = 0$ the VRPCD is reduced to two independent

CVRP. On the other hand, if c_i figures are much larger than routing costs (which depend on the Euclidean distances), optimal solutions are likely to avoid loads changing at CD. As our computational results demonstrate, even for the largest values of c_i considered here, optimal solutions include loads changing at CD.

As discussed before, the performance of DP based column generation algorithms depends upon resource constraints. As more constrained is the problem, less labels can be generated and extended by DP algorithms on solving pricing problems, which allows such algorithms to converge faster. The original instances in Wen et al. [2009] are strictly constrained on capacity, favoring the application of DP algorithms on column generation. However, we are interested here in evaluating our algorithms in a more comprehensive routing scenario. For this reason, we generate another set of instances, containing extended capacities. To be more specific, for each instance of our test set (described above) we generate another, less restricted, by doubling the capacity of vehicles and halving (rounded up) the number of vehicles of such instances. Hereinafter, we refer to the set of instances with original capacity as set 1, while those instances with the extended capacity is referred as set 2. All instances considered here are available for download at `http://www.dcc.ufmg.br/~fsantos/instances/`.

All of the algorithms described in this chapter were coded in `C++`. We also used the `CPLEX` (release 12.1) as the LP and MIP solver. Computational experiments were conducted with an Intel Core 2 Quad machine running at 2.2 GHz, with 4 Gbytes of RAM memory, under Linux Operating System. A time limit of four hours was imposed on the execution of the algorithms, except if we state otherwise on the text.

Let us start our experiments by evaluating the performance of the GRASP heuristic algorithm, used to price out routes on BP#2. We first evaluate how path-relinking helps on the intensification of solutions. To that end, we select the instance $20a$ (with $n = 20$ requests), added negative reduced costs on its vertices (randomly chosen from the interval $[-50, 0]$) and solve it using the DP algorithm. Then, we measure the CPU times (in seconds) the GRASP heuristic needs to achieve the optimal solution returned by DP, turning path-relinking ON and OFF. We report the results on a TTT (time-to-target) plot [Aiex et al., 2005] in Figure 3.2, which shows for a given time (in seconds) the probability to the algorithm evaluate the optimal solution.

According to the Figure 3.2, GRASP heuristic converges faster to optimal solutions when the path-relinking procedure is applied. For the proposed experiment, after running for 2 seconds the probability of GRASP heuristic to find the optimal solution is about 95% and 60% turning path-relinking ON and OFF, respectively. For that reason, we always run the GRASP algorithm using the path-relinking procedure.

The next experiment is proposed to evaluate the performance of the GRASP

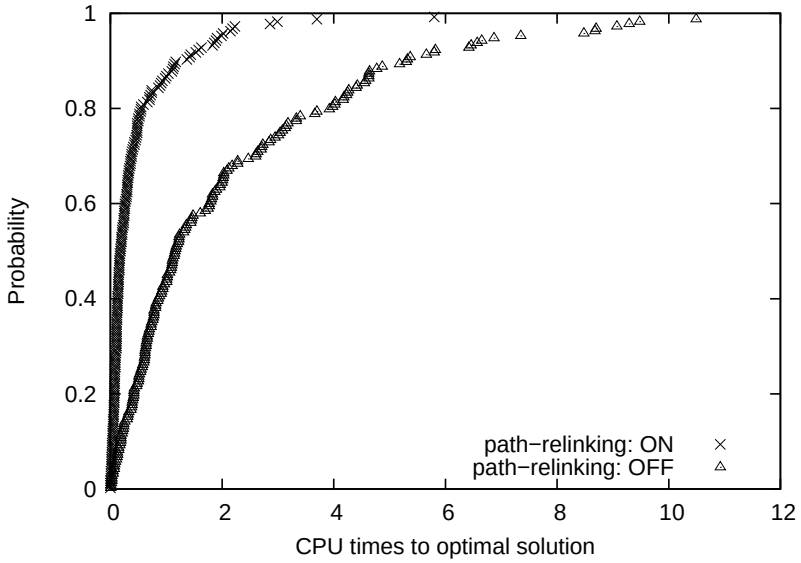


Figure 3.2: How path-relinking helps on the intensification of solutions of GRASP heuristic

heuristic according to its two parameters: number of iterations and α. Such an evaluation allows us to define the best setting of parameters for the heuristic execution. We report on table 3.1 the average results of 50 runs of the heuristic on solving those instances with $n = 30$ requests (61 vertices) of set 1. We added a negative cost (randomly defined) to the arcs of the graph and the heuristic looks for negative reduced cost paths. For each instance, we show the shortest path cost found by the heuristic and the CPU time to evaluate it for a given number of iterations, which varies from 1000 to 20000. The parameter α is fixed as 0.5 for all algorithm executions. In the last column of table 3.1 we report the optimal path value evaluated using the DP algorithm.

	1000 it		10000 it		20000 it		optimal
instance	cost	time(s)	cost	time(s)	cost	time(s)	cost
30a	208.93	0.42	201.68	4.27	199.96	8.71	193.23
30b	90.19	0.30	73.95	3.39	72.81	7.03	61.73
30c	10.16	0.26	7.62	3.01	7.40	6.08	-70.77
30d	240.93	0.36	204.22	4.00	188.12	8.19	172.12
30e	-328.25	0.29	-340.76	3.52	-343.70	8.02	-355.92
average	44.39	0.32	29.34	3.63	24.91	7.60	0.07

Table 3.1: Performance of the GRASP heuristic according to the number of iterations

Results from table 3.1 show that the quality of solutions improves along the iterations while the execution time increases. Near optimal pricing solutions can be evaluated faster using the heuristic algorithm. Moreover, for some executions the

heuristic is able to achieve the optimal pricing value for the instances $30a$, $30b$, $30d$ and $30e$ when the number of iterations is 10000 or greater. Although the heuristic achieves better results for 10000 iterations or more, we decided to fix the number of iterations in 1000 due to efficiency reasons. The BP algorithm of section 3.3 invoke the heuristic algorithm almost a hundred times in order to evaluate the LPMP for some instances. Thus, the performance of the heuristic impacts directly on the BP performance.

In Figure 3.3 we depict the quality of the heuristic solutions in terms of parameter α when the heuristic runs over 1000 iterations. On axis x, α starts with value 0 and ends with value 1 using step 0.1, while axis y presents the distance of the solutions from the optimal value in percentual figures. Over again, we report the average results of 50 executions of the heuristic.

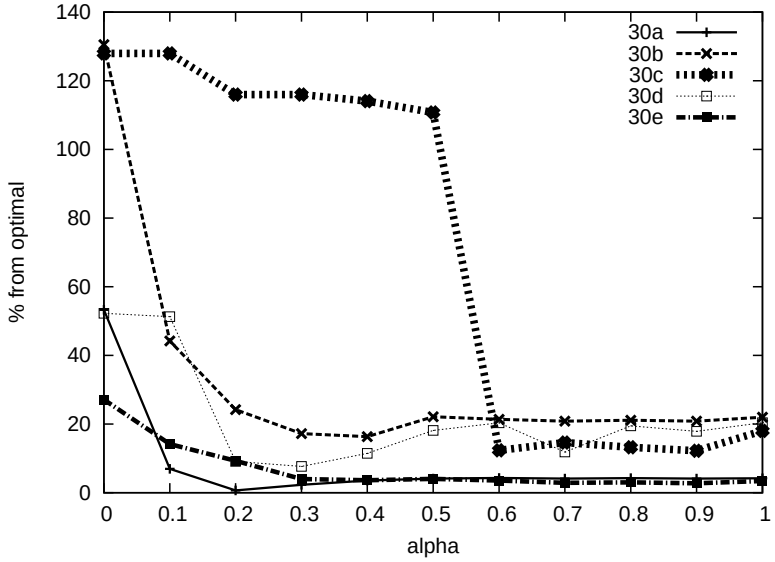


Figure 3.3: Evaluation of parameter α on heuristic behaviour

The parameter α controls the greediness to build solutions in the construction phase of GRASP heuristics. According to Figure 3.3, to build solutions greedly ($\alpha = 0$) is the worst strategy. On the other hand, to build such solutions randomly ($\alpha = 1$) allows the heuristic to achieve average results within 20% from the optimal values. However, the algorithm performs better with α on the interval $[0.2, 0.6]$ and for this reason on the BP execution it must choose randomly a value for α within this interval before each iteration.

As detailed in section 3.3.4, BP#2 makes use of two pricing approaches. One of that evaluates negative routes with the DP algorithm introduced in section 3.3.3.1, while the other prices out such routes calling GRASP and BC algorithms, described respectively in sections 3.3.3.3 and 3.3.3.2. We conducted the next experiment to

evaluate how such approaches help BP#2 to evaluate solutions. To that end, both pricing approaches were used on BP#2 to evaluate the LP bounds of model (3.16)-(3.20) for all instances of our test. Results on table 3.2 show for the first two columns informations about the instance, after that we report on columns 3-7 the LP bound followed by the number of pricing problems solved and the CPU time (in seconds) to evaluate the LP bound for instances of set 1 (on which vehicles have the original capacity). Similar entries are given on columns 8-12 for those instances with extended capacity of set 2. For such experiments we fixed the loading/unloading costs as $c_i = 20$ for all requests. Whenever BP#2 is not able to evaluate the LP bound of a given instance within a time limit of 14400 seconds, a symbol '-' is used on the table.

		set 1					set 2				
			DP		BC + GRASP			DP		BC + GRASP	
n	id	LP	# it	time(s)	# it	time(s)	LP	# it	time(s)	# it	time(s)
	a	1749.7	29	0.5	72	12	1073.9	30	2.1	110	11
	b	2147.2	28	0.4	62	16	1335.9	35	1.5	99	24
10	c	1973.0	22	0.3	63	8.4	1288.5	26	1.4	98	11
	d	1830.7	20	0.3	70	10	1159.6	18	2	101	11
	e	2027.1	20	0.4	57	7.9	1348.7	30	2.4	99	11
	a	2714.0	28	0.7	82	112	1608.1	51	113	143	165
	b	3008.1	29	1.1	85	123	1724.4	39	41	167	134
15	c	2822.3	27	0.9	96	89	1664.9	36	123	144	88
	d	2722.2	28	1.4	89	135	1636.9	37	438	166	147
	e	2947.6	34	0.9	90	93	1752.0	35	58	147	137
	a	3320.8	44	4.3	121	1658	2091.5	-	-	189	881
	b	3449.5	42	2.9	124	1768	2146.0	-	-	185	1217
20	c	3196.1	51	4.1	121	1823	2057.0	-	-	215	824
	d	3449.0	42	2.2	115	2797	2134.7	-	-	210	1764
	e	3162.9	34	8.6	153	630	2037.6	-	-	268	691
	a	4232.3	42	12	156	12981	2532.6	-	-	229	7553
	b	4436.4	40	23	109	14102	2649.4	-	-	227	11900
25	c	3850.1	53	19	163	13817	2314.6	-	-	223	5329
	d	4158.0	44	3.8	123	12988	2527.9	-	-	230	14316
	e	4282.5	41	34	157	10018	2636.8	-	-	283	8583
	a	4799.3	37	5.4	-	-	N/A	-	-	-	-
	b	4511.9	67	60	-	-	N/A	-	-	-	-
30	c	4949.8	27	6.5	-	-	N/A	-	-	-	-
	d	4391.7	40	236	-	-	N/A	-	-	-	-
	e	4731.5	55	35	-	-	N/A	-	-	-	-

Table 3.2: Performance of BP#2 for different pricing approaches for instances of sets 1 and 2

As one should expect, BP#2 evaluates the LP bounds of instances of set 1 faster when the DP algorithm is used to solve pricing problems. According to the results of table 3.2, BP#2 is capable to evaluate such bounds for any instance of set 1 with at most 236 seconds using DP as pricing algorithm, while if we set up BC + GRASP to

solve pricing problems on BP#2, the LP bounds for 4 out of 25 instances could not be evaluated within the time limit. However, for instances of set 2, BP#2 performs differently. The pricing problem resulting from such instances are more difficult to solve by DP algorithms, since more combinations are allowed for each vehicle, which holds a larger capacity. Thus, BP#2 performs poorly using DP. It is not able to evaluate the LP bounds for instances with more than 15 requests within the time limit. Whether BP#2 uses BC + GRASP, the LP bounds of instances with up to 25 requests could be evaluated. However, such bounds could not be evaluated for instances containing 30 requests for none of the approaches (such entries are labeled as 'N/A' on the table). It is clear that DP performs better than BC + GRASP to price out routes for instances strictly constrained, while for those loose constrained instances BC + GRASP outperforms DP. For that reason, in the next experiments we set up BP#2 to use DP as pricing algorithm for instances of set 1 and GRASP + BC is going to be used to price out routes for instances of set 2.

In what follows, we conduct experiments to compare the performance of BP#1 and BP#2 on solving VRPCD instances at optimality. In addition, we used the Linear Programming Based Branch-and-bound (LPBB) algorithm of `CPLEX` to solve the single commodity network flow model (2.49)-(2.60), introduced in chapter 2 using up to 4 parallel threads. A time limit of 14400 seconds was imposed on the algorithms execution.

We report those results for instances of set 1 on tables 3.3-3.5 in the following. We first show on table 3.3 the results obtained by the algorithms when loading/unloading costs are neglected ($c_i = 0$), for all requests. After that, we fix loading/unloading costs as $c_i = 20$ and $c_i = 40$ and report the results on tables 3.4 and 3.5, respectively. For each table row we report on columns 1 and 2 the instance information (n and id), while columns 3-6 illustrate the LP bound, followed by the best lower (BLB) and upper (BUB) bounds evaluated by BP#1 within the time limit and the associated duality gap, evaluated as $\frac{100BUB-BLB}{BUB}$. Similar entries are given on columns 7-10 for BP#2 and on columns 11-13 for LPBB (the LP bounds of network flow model were omitted on tables due to space limitation).

BP#1 and BP#2 perform similarly whether loading/unloading costs are neglected. Both algorithms are able to evaluate good quality BLB and BUB (small duality gaps) within the time limit, but fewer instances could be solved to proven optimality. The LP bounds provided by model (3.1)-(3.8) (BP#1) and (3.16)-(3.20) (BP#2) are the same for all instances. However, they are much stronger than those bounds evaluated by LPBB, which also evaluates loose BLB and BUB within the time limit.

		BP#1				BP#2				LPBB		
n	id	LP	BLB	BUB	gap(%)	LP	BLB	BUB	gap(%)	BLB	BUB	gap(%)
	a	**1659.9**	**1659.9**	**1659.9**	-	**1659.9**	**1659.9**	**1659.9**	-	**1659.9**	**1659.9**	-
	b	2064.3	**2130.1**	**2130.1**	-	2064.3	**2130.1**	**2130.1**	-	**2130.1**	**2130.1**	-
10	c	1900.8	**1921.9**	**1921.9**	-	1900.8	**1921.9**	**1921.9**	-	**1921.9**	**1921.9**	-
	d	1748.0	**1776.9**	**1776.9**	-	1748.0	**1776.9**	**1776.9**	-	**1776.9**	**1776.9**	-
	e	1942.9	**1950.5**	**1950.5**	-	1942.9	**1950.5**	**1950.5**	-	**1950.5**	**1950.5**	-
	a	**2594.0**	**2594.0**	**2594.0**	-	**2594.0**	**2594.0**	**2594.0**	-	2524.2	2594.0	2.6
	b	2892.7	2903.4	2929.6	0.89	2892.7	2911.0	2929.6	0.63	2693.3	2929.6	8.0
15	c	2692.9	2703.7	2742.3	1.14	2692.9	2722.4	2742.3	0.72	2489.3	2742.3	9.2
	d	2622.4	2628.8	2679.7	1.9	2622.4	2636.7	2679.6	1.60	2465.4	2677.3	7.9
	e	2818.4	2821.4	2852.8	1.1	2818.4	**2852.8**	**2852.8**	-	2674.3	2852.8	6.2
	a	3172.7	3173.2	3202.0	0.9	3172.7	3173.2	3202.9	0.9	2873.5	3260.9	11.8
	b	3312.5	3313.2	3376.4	1.87	3312.5	3325.3	3383.7	1.7	2605.1	3376.3	22.8
20	c	3062.9	3062.9	3101.5	1.25	3062.9	3067.0	3105.3	1.2	2768.9	3101.9	10.7
	d	3287.2	3287.2	3450.8	4.74	3287.2	3311.0	3358.3	1.4	2771.4	3401.0	18.5
	e	3007.7	3007.7	3054.8	1.54	3007.7	3014.8	3065.9	1.6	2835.1	3046.3	6.9
	a	4049.7	4049.7	4071.2	0.53	4049.7	4055.9	4060.0	0.1	3415.0	4325.7	21.0
	b	4234.0	4234.0	4320.5	2.0	4234.0	4243.7	4302.8	1.3	3649.8	4666.0	21.7
25	c	3658.1	3658.1	3730.8	1.95	3658.1	3668.7	3714.4	1.2	3230.8	3947.3	18.1
	d	3982.3	3982.3	4012.4	0.75	3982.3	3986.5	4003.3	0.4	3298.4	4273.1	22.8
	e	4127.2	4127.2	4242.5	2.72	4127.2	4130.7	4191.8	1.4	3520.2	5124.4	31.3
	a	4621.0	4621.0	4628.1	0.15	4621.0	4624.7	4628.1	0.07	3675.2	N/A	N/A
	b	4321.0	4321.0	4389.9	1.57	4321.0	4323.5	4382.6	1.3	3618.1	5862.6	38.2
30	c	4725.1	4725.1	4752.0	0.57	4725.1	4731.2	4752.3	0.4	3869.3	6408.1	39.6
	d	4178.6	4178.6	4209.3	0.73	4178.6	4184.7	4280.3	2.2	3527.3	4820.0	26.8
	e	4550.9	4550.9	4653.0	2.19	4550.9	4586.0	4695.2	2.3	3623.5	5688.0	36.3

Table 3.3: Computational results for set 1 neglecting loading/unloading costs at the CD ($c_i = 0, \forall p_i \in P$)

An important remark on the symmetry of model (3.1)-(3.8) can be done by looking for the LP bounds provided by BP#1 on table 3.4. Such bounds are exactly the same of those on table 3.3, evaluated for instances on which the loading/unloading costs are neglected ($c_i = 0$). Differently from BP#1, the LP bounds evaluated by BP#2 are stronger for $c_i = 20$, allowing BP#2 to provide more optimality certificates (14 out of 25) than BP#1 (7 out of 25). Over again, BP#1 and BP#2 outperform LPBB on evaluating best lower and upper bounds within the time limit.

Results of table 3.5 emphasizes that BP#2 dominates BP#1 on solving instances of set 1 at optimality. 17 instances could be solved to proven optimality by BP#2, while BP#1 is able to evaluate only 6 optimal solutions. Noteworthy is the fact that, for a given instance, whatever is the loading/unloading cost, BP#1 evaluates always the same LP bound, which difficult its convergence to optimal solutions.

LPBB compares poorly with BP#1 and BP#2. It is capable of finding optimal solutions only for instances with 10 requests. For those instances on which the algorithm is not able to find optimal solutions within the time limit, duality gaps of up to

		BP#1				BP#2				LPBB		
n	id	LP	BLB	BUB	gap(%)	LP	BLB	BUB	gap(%)	BLB	BUB	gap(%)
10	a	1659.9	**1749.7**	**1749.7**	-	**1749.7**	**1749.7**	**1749.7**	-	**1749.7**	**1749.7**	-
	b	2064.3	**2186.9**	**2186.9**	-	2147.2	**2186.9**	**2186.9**	-	**2186.9**	**2186.9**	-
	c	1900.8	**1977.5**	**1977.5**	-	1973.0	**1977.5**	**1977.5**	-	**1977.5**	**1977.5**	-
	d	1748.0	**1843.5**	**1843.5**	-	1830.7	**1843.5**	**1843.5**	-	**1843.5**	**1843.5**	-
	e	1942.9	**2044.8**	**2044.8**	-	2027.1	**2044.8**	**2044.8**	-	**2044.8**	**2044.8**	-
15	a	2594.0	**2714.0**	**2714.0**	-	**2714.0**	**2714.0**	**2714.0**	-	2557.0	2750.5	7.0
	b	2892.7	**3013.8**	**3013.8**	-	3008.1	**3013.8**	**3013.8**	-	2764.1	3013.8	8.3
	c	2692.9	2838.3	2875.5	1.29	2822.3	2868.4	2875.5	0.2	2611.1	2884.9	9.5
	d	2622.4	2727.0	2756.2	1.06	2722.2	**2756.2**	**2756.2**	-	2448.7	2777.6	11.8
	e	2818.4	2959.5	2970.2	0.36	2947.6	**2970.2**	**2970.2**	-	2819.1	2970.2	5.1
20	a	3172.7	3253.2	3387.0	3.95	3320.6	3341.9	3386.9	1.3	2909.5	3589.8	18.9
	b	3312.5	3399.6	3581.4	5.08	3449.5	**3476.7**	**3476.7**	-	2546.7	4019.7	36.6
	c	3062.9	3136.8	3264.5	3.91	3196.1	**3211.9**	**3211.9**	-	2679.3	3397.6	21.1
	d	3287.2	3390.7	3579.1	5.26	3449.0	3485.9	3530.6	1.2	2636.9	3953.8	33.3
	e	3007.7	3089.9	3192.4	3.21	3162.9	**3170.8**	**3170.8**	-	2772.8	3427.4	19.1
25	a	4049.7	4096.2	4293.8	4.6	4232.3	4240.0	4251.0	0.2	3439.3	5159.1	33.3
	b	4234.0	4212.7	4490.5	6.2	4435.9	4445.9	4461.5	0.3	3634.9	5074.4	28.3
	c	3658.1	3696.4	3988.9	7.3	3850.1	**3860.7**	**3860.7**	-	3121.2	4494.3	30.5
	d	3982.3	4051.2	4231.4	4.2	4158.8	**4175.5**	**4175.5**	-	3301.2	4859.3	32.0
	e	4127.2	4163.1	4431.0	6.0	4282.5	4295.7	4352.8	1.31	3534.4	5125.2	31.0
30	a	4621.0	4648.8	4857.0	4.2	4786.2	4789.4	4791.1	0.04	3673.3	N/A	N/A
	b	4321.0	4365.0	4651.0	6.1	4508.7	4526.7	4576.1	1.1	3616.8	6530.7	44.6
	c	4725.1	4777.4	5063.4	5.6	4939.0	**4964.3**	**4964.3**	-	3863.8	7531.1	48.7
	d	4178.6	4216.8	4560.8	7.5	4390.7	4396.3	4454.7	1.3	3594.9	6571.2	45.3
	e	4550.9	4583.3	4948.7	7.3	4731.5	4752.1	4871.9	2.4	3605.3	7534.2	52.1

Table 3.4: Computational results concerning loading/unloading costs as $c_i = 20$, $\forall p_i \in P$ for instances of set 1

53.1% were evaluated.

In overall, results on tables 3.3-3.5 show us that both BP algorithms introduced in this chapter performs good on solving instances of set 1. BP#2 outperforms BP#1 because it evaluates stronger LP bounds, allowing it to provide optimality certificates for more instances or to achieve smaller duality gaps. BP#1 is plagued with symmetry problems, which complicates the algorithm convergence, but it still performs better than LPBB.

Now, we are going to introduce results for instances of set 2. As discussed before, such instances differ from instances of set 1 only by the capacity (doubled) and the number of vehicles (halved). We report on tables 3.6-3.8 results of BP#1, BP#2 and LPBB on solving instances of set 2 concerning loading/unloading costs $c_i = 0, 20, 40$, respectively. The entries of tables are the same of that for set 1. The only difference is a symbol $*$, which is used to indicate whether the LP bound of a given instance could not be evaluated within the time limit by BP#2. In such cases, we use a Lagrangian Relaxation lower bound [Huisman et al., 2005] (an estimate of the true lower bounds),

		BP#1				BP#2				LPBB		
n	id	LP	BLB	BUB	gap(%)	LP	BLB	BUB	gap(%)	BLB	BUB	gap(%)
	a	1659.9	**1809.8**	**1809.8**	-	**1809.8**	**1809.8**	**1809.8**	-	**1809.8**	**1809.8**	-
	b	2064.3	**2226.9**	**2226.9**	-	2208.5	**2226.9**	**2226.9**	-	**2226.9**	**2226.9**	-
10	c	1900.8	**2017.5**	**2017.5**	-	**2017.5**	**2017.5**	**2017.5**	-	**2017.5**	**2017.5**	-
	d	1748.0	**1890.4**	**1890.4**	-	1888.0	**1890.4**	**1890.4**	-	**1890.4**	**1890.4**	-
	e	1942.9	**2124.0**	**2124.0**	-	2098.6	**2124.0**	**2124.0**	-	**2124.0**	**2124.0**	-
	a	2594.0	2784.4	2802.2	0.64	**2802.2**	**2802.2**	**2802.2**	-	2602.0	2802.2	7.1
	b	2892.7	**3073.8**	**3073.8**	-	**3073.8**	**3073.8**	**3073.8**	-	2928.7	3073.8	4.7
15	c	2692.9	2896.9	2938.5	1.41	2909.8	**2938.5**	**2938.5**	-	2593.0	2968.8	12.6
	d	2622.4	2768.0	2796.2	1.01	2786.3	**2796.2**	**2796.2**	-	2528.3	2796.2	9.6
	e	2818.4	3017.6	3027.9	0.34	3016.0	**3027.9**	**3027.9**	-	2844.6	3027.9	6.0
	a	3172.7	3272.1	3537.2	7.49	3413.7	3430.3	3454.4	0.7	2919.6	3459.9	15.6
	b	3312.5	3417.5	3675.3	7	3515.5	**3526.5**	**3526.5**	-	2830.3	3526.5	19.7
20	c	3062.9	3154.4	3417.0	7.7	3281.5	**3295.3**	**3295.3**	-	2787.2	3307.1	15.7
	d	3287.2	3421.5	3821.1	10.4	3537.6	3573.9	3576.5	0.1	2849.6	3576.5	20.3
	e	3007.7	3110.5	3370.5	7.7	**3237.1**	**3237.1**	**3237.1**	-	2811.0	3237.1	13.1
	a	4049.7	4100.0	4613.3	11.1	4325.0	**4334.9**	**4334.9**	-	3446.7	5180.0	33.4
	b	4234.0	4331.9	4866.0	10.9	4504.6	4515.1	4521.5	0.1	3715.4	4623.5	19.6
25	c	3658.1	3696.4	4098.0	9.8	3905.1	**3920.7**	**3920.7**	-	3233.2	4350.5	25.6
	d	3982.3	4069.6	4444.2	8.4	**4204.0**	**4204.0**	**4204.0**	-	3358.5	4217.3	20.3
	e	4127.2	4159.4	4598.1	9.5	4367.4	4375.9	4462.0	1.9	3537.0	4517.7	21.7
	a	4621.0	4649.7	5068.1	8.2	4859.1	**4864.1**	**4864.1**	-	3700.0	6712.8	44.8
	b	4321.0	4382.5	5018.5	12.6	4577.5	4580.5	4619.2	0.8	3638.4	6735.5	45.9
30	c	4725.1	4766.2	5150.7	7.4	5028.1	5038.6	5045.0	0.1	3915.1	7210.7	45.7
	d	4178.6	4217.6	4782.1	11.8	4477.5	4478.2	4504.2	0.6	3604.1	6107.4	41.0
	e	4550.9	4582.4	5242.8	12.6	4834.6	4847.1	4961.2	2.3	3629.9	7740.9	53.1

Table 3.5: Computational results concerning loading/unloading costs as $c_i = 40$, $\forall p_i \in P$ for instances of set 1

which is evaluated using the negative reduced cost of routes priced along the column generation.

According to the results of table 3.6, BP#1 outperforms BP#2 whenever loading/unloading costs are neglected ($c_i = 0$). It is able to provide optimality certificates for 8 instances against 3 from BP#2 and also evaluates smaller duality gaps at termination for those instances left unsolved. Even though models (3.1)-(3.8) and (3.16)-(3.20) provide the same LP bounds for loading/unloading values $c_i = 0$, BP#1 presents an advantage compared with BP#2 for dealing with instances loose constrained. BP#1 separates suppliers and customers into two different (smaller) subproblems, allowing pricings to be evaluated faster. BP#2 is not able to evaluate the LP bounds from model (3.16)-(3.20) for instances with more than 20 requests, due to the increasing on the complexity of subproblems. BP#1 also outperforms LPBB concerning lower and upper bounds, whereas BP#2 provides stronger lower bounds than LPBB, but poor upper bounds.

It is clear from the results of tables 3.7-3.8 that loose constrained instances of

		BP#1				BP#2				LPBB		
n	id	LP	BLB	BUB	gap(%)	LP	BLB	BUB	gap(%)	BLB	BUB	gap(%)
	a	1008.3	1033.9	1033.9	-	1008.3	1033.9	1033.9	-	1033.9	1033.9	-
	b	1275.9	1302.3	1302.3	-	1275.9	1287.8	1354.9	4.9	1302.3	1302.3	-
10	c	1202.8	1208.5	1208.5	-	1202.8	1208.5	1208.5	-	1208.5	1208.5	-
	d	1093.0	1093.0	1093.0	-	1093.0	1093.0	1093.0	-	1093.0	1093.0	-
	e	1262.3	1262.3	1262.3	-	1262.3	1262.3	1262.3	-	1262.3	1262.3	-
	a	1497.3	1514.8	1514.8	-	1497.3	1497.3	1514.8	1.1	1462.7	1514.8	3.4
	b	1629.8	1636.5	1636.5	-	1629.8	1632.2	1636.5	0.2	1600.5	1636.5	2.2
15	c	1551.9	1560.4	1560.4	-	1551.9	1553.5	1561.5	0.5	1503.4	1560.4	3.6
	d	1548.6	1569.8	1569.8	-	1548.6	1552.6	2978.1	47.8	1526.9	1569.8	2.7
	e	1651.0	1665.9	1665.9	-	1651.0	1652.5	2904.1	43.1	1565.5	1665.9	6.0
	a	1950.1	1954.0	1974.3	1.0	1950.1	1950.1	3622.9	46.1	1829.7	1974.3	7.3
	b	2004.2	2006.2	2252.6	10.9	2004.2	2004.2	3663.0	45.2	1734.8	2047.5	15.2
20	c	1925.8	1929.6	1949.0	1.0	1925.8	1925.8	3444.3	44.0	1777.2	1963.2	9.4
	d	1998.5	2000.2	2010.9	0.5	1998.5	1998.5	3845.5	48.0	1834.7	2010.0	8.7
	e	1902.6	1902.6	1953.5	2.6	1902.6	1902.9	3845.1	50.5	1822.3	1937.1	5.9
	a	2369.9	2369.9	2480.5	4.4	2351.8_*	2351.8_*	4584.3	48.7	2059.3	2396.0	14.5
	b	2445.3	2448.7	2508.6	2.3	2436.8_*	2436.8_*	4126.0	40.9	2241.4	2483.8	9.7
25	c	2134.6	2134.6	2144.1	0.4	2130.8_*	2130.8_*	4196.0	49.2	1949.2	2144.1	9.0
	d	2365.3	2365.3	2490.4	5.0	2312.8_*	2312.8_*	4155.1	44.3	2064.8	2418.5	14.6
	e	2483.9	2483.9	2529.8	1.8	2476.8_*	2476.8_*	4958.3	50.0	2330.5	2527.0	7.7
	a	2533.2	2533.2	2611.8	3.0	1864.9_*	1864.9_*	4798.5	61.1	1963.6	3019.5	34.9
	b	2331.5	2331.6	2440.7	4.4	2168.1_*	2168.1_*	4470.3	51.5	1900.6	2368.2	19.7
30	c	2580.9	2580.9	2586.6	0.2	2429.9_*	2429.9_*	5712.7	57.4	2166.4	2585.7	16.2
	d	2377.9	2377.9	2469.2	3.7	2253.3_*	2253.3_*	4841.6	53.4	1989.1	2756.0	27.8
	e	2628.3	2628.3	3023.4	13.0	2405.2_*	2405.2_*	5121.6	53.0	2022.7	2723.2	25.7

Table 3.6: Computational results for instances with extended capacity (set 2). Loading/unloading costs at CD are neglected ($c_i = 0$, $\forall p_i \in P$)

set 2 are harder to solve using BP algorithms. To be more specific, BP#1 and BP#2 provides optimality certificates for respectively 19 and 32 instances of set 1 with loading/unloading costs $c_i = \{20, 40\}$ and attains an average duality gaps of respectively 7.6% and 0.8% for those instances left unsolved within time limit. For the same group of instances on set 2, BP#1 and BP#2 are respectively capable to evaluate 20 and 19 optimal solutions and average duality gaps at termination are of 20.4% and 32.3%. LPBB performs similarly for both set of instances, since it not depends of solving subproblems for evaluating its bounds.

Roughly speaking, both BP algorithms perform reasonably good in evaluating lower bounds for instances of set 2. Upper bounds from LPBB are usually sharper, due to embed heuristics from `CPLEX` running in parallel using up to 4 threads. If we combine the best values of BLB and BUB attained by the three algorithms, the average duality gap may be reduced to 6.5%.

		BP#1				BP#2				LPBB		
n	id	LP	BLB	BUB	gap(%)	LP	BLB	BUB	gap(%)	BLB	BUB	gap(%)
	a	1008.3	1073.9	1073.9	-	1073.9	1073.9	1073.9	-	1073.9	1073.9	-
	b	1275.9	1344.3	1344.3	-	1335.9	1344.3	1344.3	-	1344.3	1344.3	-
10	c	1202.8	1288.5	1288.5	-	1288.5	1288.5	1288.5	-	1288.5	1288.5	-
	d	1093.0	1159.6	1159.6	-	1159.6	1159.6	1159.6	-	1159.6	1159.6	-
	e	1262.3	1357.7	1357.7	-	1348.7	1357.7	1357.7	-	1357.7	1357.7	-
	a	1497.3	1633.2	1633.2	-	1608.1	1623.3	1633.2	0.6	1552.0	1633.2	4.9
	b	1629.8	1724.4	1724.4	-	1724.4	1724.4	1724.4	-	1724.4	1724.4	-
15	c	1551.9	1664.9	1664.9	-	1664.9	1664.9	1664.9	-	1664.9	1664.9	-
	d	1548.6	1636.9	1636.9	-	1636.9	1636.9	1636.9	-	1615.0	1636.9	1.3
	e	1651.0	1767.2	1767.2	-	1752.0	1757.0	1767.7	0.6	1737.9	1767.2	1.6
	a	1950.1	2022.4	2196.8	7.9	2091.5	2094.0	3908.4	46.4	1901.7	2174.7	12.5
	b	2004.2	2066.6	2213.3	6.6	2146.0	2146.8	3634.9	40.9	1799.8	2194.5	18.0
20	c	1925.8	1988.5	2135.1	7.1	2057.0	2060.8	3836.4	46.2	1850.0	2081.4	11.1
	d	1998.5	2097.8	2147.4	2.3	2134.7	2134.7	2134.7	-	1941.0	2134.7	9.0
	e	1902.6	1987.9	2196.5	9.5	2037.6	2039.6	2046.6	0.3	1922.7	2046.6	6.0
	a	2369.9	2400.6	2686.0	10.6	2532.6	2532.6	4585.4	44.7	2003.1	2587.5	22.5
	b	2445.3	2524.7	2784.5	9.3	2649.4	2649.4	4190.7	36.7	2254.4	2694.8	16.3
25	c	2134.6	2183.3	2393.0	8.7	2314.6	2314.6	4680.5	50.5	1979.4	2331.5	15.1
	d	2365.3	2413.5	2663.2	9.3	2527.9	2527.9	4321.6	41.5	2116.7	2711.8	21.9
	e	2483.9	2501.4	2759.7	9.3	2636.8	2636.8	4512.3	41.5	2253.3	2719.6	17.1
	a	2533.2	2570.7	5250.0	51.0	2262.6_*	2262.6_*	5046.0	55.1	1970.1	3588.7	45.1
	b	2331.5	2349.3	2786.0	15.6	2358.3_*	2358.3_*	4569.7	48.3	1939.5	2596.5	25.3
30	c	2580.9	2611.5	2880.4	9.3	2326.6_*	2326.6_*	5567.9	58.2	2216.1	2844.0	22.0
	d	2377.9	2378.2	2856.7	16.7	2188.5_*	2188.5_*	4659.9	53.0	2022.3	3031.7	33.2
	e	2628.3	2643.4	3261.2	18.9	2437.8_*	2437.8_*	5614.7	56.5	2046.1	2948.5	30.6

Table 3.7: Solving instances of set 2 for loading/unloading costs as $c_i = 20$, $\forall p_i \in P$

3.5 Concluding Remarks

We presented in this chapter exact solution approaches for solving the VRPCD, a recent problem that integrates routing and scheduling decisions in supply chains. We introduced two BP algorithms with different structures labeled BP#1 and BP#2. BP#1 decomposes the VRPCD into two CVRP, which leads to smaller and easier pricing problems. However it presents as shortcoming a deep symmetry. BP#2 solves symmetry problems of BP#1, but has a pricing problem quite complicated. For that reason, we implemented different algorithms for solving pricing problems: Dynamic Programming, Branch-and-cut and GRASP heuristic.

We evaluated the performance of our algorithms using different data sets. On the one hand, BP#2 dominates BP#1 for solving strictly constrained instances (named instances of set 1). On the other hand, BP#1 performs better than BP#2 when instances are loose constrained (instances of set 2). In overall, both BP approaches outperformed those obtained from a Linear Programming Branch-and-bound based approach from the MIP package of `CPLEX` running with up to 4 parallel threads, solving

		BP#1				BP#2				LPBB		
n	id	LP	BLB	BUB	gap(%)	LP	BLB	BUB	gap(%)	BLB	BUB	gap(%)
	a	1008.3	1113.9	1113.9	-	1113.9	1113.9	1113.9	-	1113.9	1113.9	-
	b	1275.9	1384.3	1384.3	-	1378.5	1384.3	1384.3	-	1384.3	1384.3	-
10	c	1202.8	1357.2	1357.2	-	1340.1	1357.2	1357.2	-	1357.2	1357.2	-
	d	1093.0	1171.4	1171.4	-	1171.4	1171.4	1171.4	-	1171.4	1171.4	-
	e	1262.3	1405.0	1405.0	-	1399.4	1405.0	1405.0	-	1405.0	1405.0	-
	a	1497.3	1713.2	1713.2	-	1678.0	1701.0	1713.2	0.7	1623.1	1713.2	5.2
	b	1629.8	1804.4	1804.4	-	1804.4	1804.4	1804.4	-	1804.4	1804.4	-
15	c	1551.9	1744.9	1744.9	-	1744.9	1744.9	1744.9	-	1744.9	1744.9	-
	d	1548.6	1676.9	1676.9	-	1676.9	1676.9	1676.9	-	1672.7	1676.9	0.2
	e	1651.0	1847.2	1847.2	-	1815.2	1820.4	1850.8	1.6	1758.7	1847.2	4.7
	a	1950.1	2029.9	2355.9	13.8	2128.9	2128.9	2128.9	-	1874.3	2202.2	14.8
	b	2004.2	2099.1	2486.1	15.5	2227.9	2229.4	3746.7	40.4	1835.0	2281.2	19.5
20	c	1925.8	2014.2	2347.1	14.1	2130.2	2133.8	2138.5	0.2	1868.9	2259.9	17.3
	d	1998.5	2122.0	2350.3	9.7	2194.7	2194.7	2194.7	-	1918.7	2240.0	14.3
	e	1902.6	1989.3	2297.8	13.4	2104.5	2108.0	2117.6	0.4	1938.9	2117.6	8.4
	a	2369.9	2407.6	2998.0	19.7	2611.6	2611.6	4503.3	42.0	2042.3	2900.7	29.5
	b	2445.3	2526.7	3056.1	17.3	2727.7	2728.4	4057.7	32.7	2285.8	2908.2	21.4
25	c	2134.6	2200.2	2633.0	16.4	2399.4	2399.4	4175.6	42.5	1941.5	2468.2	21.3
	d	2365.3	2421.5	3069.1	21.1	2572.5	2572.5	4137.2	37.8	2065.8	2632.6	21.5
	e	2483.9	2514.9	2982.1	15.6	2740.0	2740.0	4901.2	44.1	2354.9	2927.0	19.5
	a	2533.2	2568.8	3217.8	20.1	2918.8	2918.8	4698.3	37.8	1954.1	3261.3	40.0
	b	2331.5	2358.6	3134.3	24.7	1951.0_*	1951.0_*	4502.4	56.6	1919.0	2661.8	27.9
30	c	2580.9	2607.1	3198.7	18.4	3009.8	3009.8	5348.5	43.7	2164.7	4196.8	48.4
	d	2377.9	2413.7	3354.3	28.0	2330.4_*	2330.4_*	4774.2	51.1	2071.2	2752.7	24.7
	e	2628.3	2640.3	6435.3	58.9	2490.6_*	2490.6_*	5347.0	53.4	2084.9	3187.0	34.5

Table 3.8: Solving instances of set 2 for loading/unloading costs as $c_i = 40$, $\forall p_i \in P$

a network flow model for the VRPCD.

As further steps of research we may conduct a deeper study on model (3.1)-(3.8) in order to prevent symmetrical solutions and accelerate the BP#1 convergence. We can also study alternatives for solving the pricing problem of BP#2 in order to solve it faster. We attempted to implement the classical relaxation on the elementarity condition of routes, but the results were not promising. One possible explanation is that routes priced out for BP#2 include routing plus loading/unloading costs. To relax the elementarity condition did not allow us to prune much labels due to the loading/unloading costs. So the algorithm did not perform as good as expected.

Chapter 4

A Branch-and-price algorithm for the Pickup and Delivery Problem with Cross-Docking

In this chapter, we extend the VRPCD model (3.16)-(3.20) introduced in chapter 3 by allowing vehicles to perform direct routes, on which the requests are collected and delivered by the same vehicle, bypassing the CD. We call this problem as the Pickup and Delivery Problem with Cross-Docking (PDPCD). Once again, to solve the proposed formulation we implemented a BP algorithm, which is described along this chapter. In the section 4.1 we present an IP model for the PDPCD and discuss the changes introduced in the previous formulation to allow direct routes on solutions, while in sections 4.2 and 4.3 we show how the LP bounds implied by such a formulation are evaluated as well as further details about the BP implementation. We report in the section 4.4 the computational experience running the proposed BP algorithm to solve VRPCD instances and discuss the saving achieved using the direct routes approach. Finally, we conclude the chapter in the section 4.5.

4.1 Integer Programming Formulation

Let $G = (V, A)$ be a directed graph with set of vertices $V = \{0\} \cup S \cup C$, where $S = \{1, \ldots, n\}$ and $C = \{1', \ldots, n'\}$ denote, respectively, sets of n suppliers and n customers and vertex 0 represents the CD. Consider that $P = \{(i, i', q_i) : i = 1, \ldots, n\}$ denotes a set of requests with n triples, each one representing a demand (or load) $q_i > 0$ to be collected from a supplier i and delivered to a customer i'. Consider as well that

a set $\mathcal{K}$, of K homogeneous vehicles of capacity Q is available.

Define costs $\{c_{ij} \geq 0 : (i,j) \in A\}$ (satisfying the triangle inequalities) and $\{c_i : i = 1, \ldots, n\}$ to be incurred, respectively, when the arcs of G are traversed by the vehicles and when a load q_i is moved from one vehicle to another at the CD. The cost c_i is incurred only once, when load q_i is delivered by a vehicle that does not collect it. PDPCD consists of finding K routes, one for each vehicle, in order to guarantee that each load q_i will be collected from its supplier ($i \in S$) and delivered to its customer ($i' \in C$). The loads shipped in a vehicle cannot exceed the capacity Q and the goal is to minimize the transportation costs plus the sum of the costs of changing loads at the CD. Two kinds of routes are considered to meet the transportation requests:

1. *docking routes*, on which vehicles leave the CD, visit a subset of suppliers, return to the CD, implement load changes at the CD, leave the CD to visit a subset of customers. After visiting the last customer, the vehicle returns empty to the CD. Due to the triangle inequality on the arcs costs, these routes either collect loads that they do not deliver, or deliver loads that they do not collect, or both.

2. *pickup and delivery routes*, on which vehicles leave the CD, visit a subset of suppliers and after the last one is visited, start delivering the collected loads to the customers, without a stop at the CD. Only after the last customer is visited, the vehicle returns empty to the CD.

Assume that R denotes the set of *docking routes* while R_d denotes the set of *pickup and delivery routes*. For each route $r \in R \cup R_d$, let c_r denote its cost, given by the sum of the costs of the arcs traversed by the vehicle. Assume we are given a binary parameter a_{ir} indicating whether route $r \in R \cup R_d$ visits ($a_{ir} = 1$) or not ($a_{ir} = 0$) vertex $i \in V$. For a route $r \in R$, consider also a binary parameter $b_{ir} := \max\{0, a_{i'r} - a_{ir}\}$ that assumes value 1 if and only if route r visits customer i' without visiting the corresponding supplier i. In the model, we use binary decision variables λ_r and δ_r, respectively to select routes from sets R and R_d. The model also makes use of binary variables τ_i to define whether ($\tau_i = 1$) or not ($\tau_i = 0$) load q_i is moved from one vehicle to another at the CD. The IP formulation for PDPCD is:

$$\min \sum_{r \in R} c_r \lambda_r + \sum_{r \in R_d} c_r \delta_r + \sum_{i=1}^{n} c_i \tau_i \tag{4.1}$$

$$\sum_{r \in R} \lambda_r + \sum_{r \in R_d} \delta_r = K \tag{4.2}$$

$$\sum_{r \in R} a_{ir}\lambda_r + \sum_{r \in R_d} a_{ir}\delta_r = 1 \qquad \forall\, i \in V \setminus \{0\} \tag{4.3}$$

$$\tau_i - \sum_{r \in R} b_{ir}\lambda_r \geq 0 \qquad i = 1, \ldots, n \tag{4.4}$$

$$\lambda \in \mathbb{B}^{|R|}, \delta \in \mathbb{B}^{|R_d|}, \tau \in \mathbb{B}^n. \tag{4.5}$$

The objective function (4.1) minimizes the total cost, i.e., the cost of the routes plus the load changing costs. Constraint (4.2) assures that all K vehicles are used while constraints (4.3) guarantee that each supplier and customer will be visited exactly once. Inequalities (4.4) couple τ and λ variables, imposing that $\tau_i = 1$ whenever a vehicle visits customer i' without visiting supplier i. The decision variables space are defined in (4.5). It is noteworthy that the above formulation is precisely the VRPCD formulation (3.16)-(3.20) after including the decision variables $\{\delta_r : r \in R_d\}$, which model the *pickup and delivery routes* in the PDPCD.

It may be the case that some loads in a supply chain system must necessarily stop at the CD, while for other loads, stopping at the CD must be avoided. The Integer Program (4.1)-(4.5) can be easily specialized to deal with such modeling issues. To be more precise, assume that the suppliers S are partitioned into the following disjoint sets: (i) a set of suppliers $S_{CD} \subseteq S$ whose loads must stop at the CD, (ii) a set of suppliers $S_{NS} \subseteq S$ whose loads must not stop at the CD, and, (iii) a set of suppliers $S_F \subseteq S$ whose loads are neither imposed to stop nor to avoid the stop at the CD. Likewise, define the following subsets of customers: $C_{CD} := \{i' \in C : i \in S_{CD}\}$, $C_{NS} := \{i' \in C : i \in S_{NS}\}$ and $C_F := \{i' \in C : i \in S_F\}$. Accordingly, define $V_{CD} := S_{CD} \cup C_{CD}$, $V_{NS} := S_{NS} \cup C_{NS}$ and $V_F := S_F \cup C_F$. To assign routes to customers and suppliers satisfying such requirements, one just needs to replace constraints (4.3) by

$$\sum_{r \in R} a_{ir}\lambda_r = 1 \qquad \forall\, i \in V_{CD}, \tag{4.6}$$

$$\sum_{r \in R_d} a_{ir}\delta_r = 1 \qquad \forall\, i \in V_{NS}, \tag{4.7}$$

$$\sum_{r \in R} a_{ir}\lambda_r + \sum_{r \in R_d} a_{ir}\delta_r = 1 \qquad \forall\, i \in V_F. \tag{4.8}$$

4.2 Solving the Linear Programming relaxation by Column Generation

In this section, we describe how the LP bounds implied by model (4.1)-(4.5) are evaluated. Unless stated otherwise, in the exposition that follows, we assume that $V_F = C \cup S$, i.e., $V_{CD} = \emptyset$, $V_{NS} = \emptyset$. Specific details will be given when that does not apply.

Let us consider the Linear Programming Master Program (LPMP), given by (4.1)-(4.4) and

$$\lambda_r \geq 0 \qquad r \in R \tag{4.9}$$

$$\delta_r \geq 0 \qquad r \in R_d \tag{4.10}$$

$$0 \leq \tau_i \leq 1 \quad i = 1, \ldots, n. \tag{4.11}$$

Assume that α, $\{\theta_i : i \in V \setminus \{0\}\}$ and $\{\chi_i : i = 1, \ldots, n\}$ denote dual variables respectively assigned to constraints (4.2), (4.3) and (4.4). Assume as well that, by a method to be described in section 4.3, initial subsets of routes $\hat{R} \subset R$ and $\hat{R}_d \subset R_d$ ($|\hat{R}| << |R|$, $|\hat{R}_d| << |R_d|$) are available and that, a basic solution $\{\lambda_r^* : r \in \hat{R}\}$, $\{\delta_r^* : r \in \hat{R}_d\}$ and $\{\tau_i^* : i = 1, \ldots, n\}$ to the Restricted Linear Programming Master Problem (RLPMP) formulated and solved after replacing R and R_d in (4.1)-(4.4), (4.9)-(4.11), respectively by $\hat{R}$ and $\hat{R}_d$, is also available. If the constraints

$$\alpha^* + \sum_{i \in V \setminus \{0\}} a_{ir}\theta_i^* - \sum_{i=1}^{n} b_{ir}\chi_i^* \leq c^r \qquad \forall r \in R \setminus \hat{R} \tag{4.12}$$

$$\alpha^* + \sum_{i \in V \setminus \{0\}} a_{ir}\theta_i^* \leq c^r \qquad \forall r \in R_d \setminus \hat{R}_d \tag{4.13}$$

of the LPMP dual are satisfied, then $\{\lambda_r^* : r \in \hat{R}\}$, $\{\lambda_r^* = 0 : r \in R \setminus \hat{R}\}$, $\{\delta_r^* : r \in \hat{R}_d\}$, $\{\delta_r^* = 0 : r \in R_d \setminus \hat{R}_d\}$ and $\{\tau_i^* : i = 1, \ldots, n\}$ solve the LPMP. Otherwise, sets $\hat{R}$ and $\hat{R}_d$ must be enlarged with those routes that respectively violate (4.12) and (4.13). A new RLPMP is formulated and re-optimized. The process is repeated until constraints (4.12) and (4.13) are satisfied.

4.2.1 Column Generation Subproblems

The problem of pricing routes in column generation algorithms is usually formulated as Elementary Shortest Path Problems with Resource Constraints (ESPPRC) [Feillet et al., 2004; Jepsen et al., 2008], which are known to be NP-hard. In this study, we deal with two variants of the classical ESPPRC as column generation subproblems. The first subproblem was introduced before, in section 3.3.3, and consists of pricing *docking routes* considering costs for routing and loading/unloading together. As described in section 3.3.3, we implemented DP, BC and heuristic algorithms for solving such a pricing problem, which can also be directly applied to the solution of model (4.1)-(4.5) for the PDPCD.

Let us now discuss how *pickup and delivery routes* ($r \in R_d$) are priced out by a DP algorithm. The proposed algorithm is based on that introduced in Ropke and Cordeau [2009]. Starting from an initial path that includes only the CD, the algorithm builds new paths from existing ones. A label L is assigned to each path kept in a list of paths. The label stores the load $q(L)$ of the path (the sum of the loads in the vehicle, at an intermediate or at the end vertex), its end node $e(L)$ (the last vertex in C or in S visited by the path), a set $O(L) \subseteq S$ of suppliers visited by the path, whose corresponding customers were not visited by the path yet and a set of forbidden vertices $U(L) \subseteq S$. The latter is used to avoid, for example, that the vehicle visits a supplier whose load exceeds the vehicle's free capacity or to guarantee that a supplier will be visited at most once, assuring that the path elementarity condition is met. The label also stores the accumulated reduced cost $c(L)$ of the path. Assume that $\overline{S}$ and $\overline{A}$ respectively denote the set of suppliers visited by the path and the set of arcs traversed by the path. Given a path whose label is L, if $e(L) \in S$, we define $\overline{C} := \{i' \in C : i \in \overline{S}\}$. That being stated, the reduced cost of a path, $c(L)$, is defined as $c(L) := -\alpha^* + \sum_{(i,j)\in\overline{A}} c_{ij} - \sum_{i\in\overline{S}} \theta_i^* - \sum_{i'\in\overline{C}} \theta_{i'}^*$.

Here again, dominance rules are used to identify states (paths) that, being provably suboptimal, do not need to be extended. To be more specific, assume that L_1 and L_2 are two labels assigned to two paths. We say that L_1 dominates L_2 if the reduced cost of L_1 does not exceed the reduced cost of L_2 and if all feasible extensions to L_2

are also feasible to L_1. We say that label L_1 dominates L_2 if:

$$\begin{aligned} e(L_1) &= e(L_2), \\ c(L_1) &\leq c(L_2), \\ q(L_1) &\leq q(L_2), \\ O(L_1) &\subseteq O(L_2), \\ U(L_1) &\subseteq U(L_2). \end{aligned}$$

A path of label L can be extended to $j \in S$ if: $(v(L), j) \in A$, $q(L) + q_j \leq Q$ and $j \notin U(L)$. Likewise, L can be extended to $j' \in C$ if $(v(L), j) \in A$ and $j \in O(L)$. The extension to the CD can only take place if $O(L) = \emptyset$.

As discussed in the previous chapter, DP algorithms are sensitive to resource constraints. Because our DP for pricing pickup and delivery is constrained only by the vehicles' capacity, its performance may be worsen whenever such a resource is not restricted. To deal with such cases, we modify GRASP and BC algorithms described respectively in sections 3.3.3.3 and 3.3.3.2, to be able to evaluate pickup and delivery negative routes on pricing. Such a slight modification consists of enforcing that suppliers and customers for a given request are visited by the same route and also imposing precedence constraints on suppliers vertices. Both, GRASP and BC, are going to be useful to evaluate solutions for instances of set 2, which contains less vehicles with large capacity.

In the current study, we observed that algorithms implemented for pricing *pickup and delivery routes* ($r \in R_d$) work faster than those algorithms for pricing *docking routes* ($r \in R$). The tighter dominance rules and the precedence conditions imposed on routes $r \in R_d$ are the main reasons for that. Therefore, after each RLPMP is solved, *pickup and delivery routes* are always priced before *docking routes*. Since the algorithms for pricing *docking routes* is less effective than its counterpart for pricing *pickup and delivery routes*, pricing *docking routes* only takes place when no route in R_d is found to have negative reduced cost.

The problem of pricing both *docking routes* or *pickup and delivery routes* can be easily specialized for the case where $V_{NS} \neq \emptyset$ and/or $V_{CD} \neq \emptyset$. That can be accomplished by removing vertices $i \in V_{CD}$ ($i \in V_{NS}$) from the network, when pricing routes $r \in R_d$ ($r \in R$). In such cases, dual variables θ_i are meant to be assigned to constraints (4.9)-(4.11), respectively for $i \in V_{CD}, i \in V_{NS}$ and $i \in V_F$.

4.3 Further implementation details

The first column generation iteration is initialized by setting $\hat{R} = \emptyset$ and by randomly choosing $\hat{R}_d$ as follows. We generate K pickup and delivery routes, randomly choosing sets of suppliers whose sum of loads does not exceed Q. Then any order of visiting the suppliers and the customers is chosen. In doing so, we guarantee that constraint (4.2) and set partitioning constraints (4.3) are satisfied. Therefore, when the first RLPMP is solved, the values of $\{\tau_i : i = 1, \ldots, n\}$ are always zero. When $V_F \neq S \cup C$, initial solutions satisfying the vehicles' capacities are obtained by means of randomly choosing vertices in V_{CD} to compose *docking routes* and vertices in V_{NS} to build *pickup and delivery routes*. After that, if the number of routes already chosen is K, we attempt to include all vertices in V_F in those routes of R and R_d chosen. Otherwise, we create routes spanning only those vertices in V_F to complete the desired number of K routes.

Assume that $\{\lambda_r^* : r \in R\}$, $\{\delta_r^* : r \in R_d\}$ and $\{\tau_i^* : i = 1, \ldots, n\}$ denote a basic optimal solution to the LPMP. Note that whenever $\{\lambda_r^* : r \in R\}$ and $\{\delta_r^* : r \in R_d\}$ are integer, so do variables $\{\tau_i^* : i = 1, \ldots, n\}$. Therefore, whenever that applies, $\{\lambda_r^* : r \in R\}$, $\{\delta_r^* : r \in R_d\}$ and $\{\tau_i^* : i = 1, \ldots, n\}$ solve (4.1)-(4.5) as well. Otherwise, we resort to branching, as follows.

For each arc $(i, j) \in A$, we compute $x_{ij} := \sum_{r \in R} a_{ijr}\lambda_r^* + \sum_{r \in R_d} a_{ijr}\delta_r^*$, where a_{ijr} denotes a binary parameter that indicates whether or not arc (i, j) is used by route r. We branch on the most fractional arc $(p, q) \in \arg\min_{(i,j) \in A}\{\left|x_{ij} - \frac{1}{2}\right|\}$. Ties are broken randomly. Assuming that (p, q) is the arc on which we branch, two nodes are created. In one of them, we impose the constraint $\sum_{r \in R} a_{pqr}\lambda_r + \sum_{r \in R_d} a_{pqr}\delta_r = 1$ on the Linear Programming Master Program to be solved at that child node. In the other, we add the constraint $\sum_{r \in R} a_{pqr}\lambda_r + \sum_{r \in R_d} a_{pqr}\delta_r = 0$. Note that, in doing so, the structure of the pricing problems does not change.

To provide an initial basic solution when a child node is evaluated, we include an artificial column/route (with sufficiently high cost and appropriate a_{pqr} entries) and solve the associated LPMP as discussed. At the optimal solution to the LPMP implied by that node, if the artificial variable is not left out of the optimum basis (possibly after pivoting for dealing with primal degeneracy), the node is considered infeasible.

Assume now that $\tilde{R}_d \subset R_d$ and $\tilde{R} \subset R$ are the sets of routes included in the last RLPMP, formulated when the LP relaxation of model (4.1)-(4.5) is solved. In order to obtain a valid upper bound for PDPCD, we replace sets R_d and R respectively by $\tilde{R}_d$ and $\tilde{R}$ in (4.1)-(4.5). The corresponding IP is solved to optimality by `CPLEX` MIP package (release 12.1). The optimal cost to that IP is used as an upper bound for the optimal PDPCD cost. `CPLEX` is also the Linear Programming package used to solve

each RLPMP in our BP algorithm, which implements a best-first search policy.

As an attempt to speed up our algorithm, at the cost of lowering the associated LP bounds, we evaluated the effect of relaxing the elementarity condition on the paths, when solving the two pricing problems. Our computational results indicated that, for the instances considered here, the reduction in the CPU time involved in DP algorithms does not make up the increase on the number of branch-and-bound nodes. Therefore, all computational results described next were obtained without relaxing the elementarity condition.

4.4 Computational Experiments

In this section, we report our computational experience with the BP algorithm implemented for the PDPCD. The algorithm was tested using that test bed introduced for VRPCD in section 3.3, which comprises two set of instances. Instances of both sets hold the same properties: they are based on real world instances adapted from Wen et al. [2009], containing $n \in \{10, 15, 20, 25, 30\}$ requests (up to 61 vertices). However, instances of sets 1 and 2 differs by the capacity and number of vehicles (set 2 have less vehicles with larger capacity). Three values of loading/unloading costs ($c_i = 0, 20, 40$) were considered. For a given instance, the same loading/unloading costs were imposed to all goods.

The algorithms discussed here were coded in `C++`. Experiments were conducted with an Intel Core 2 Quad machine, running at 2.2 GHz, with 4 Gbytes of RAM memory, under Linux Operating System. A time limit of four CPU hours was imposed on the execution of the BP algorithms.

We first evaluate our BP algorithm for instances of set 1. In Tables 4.1-4.3, we present detailed computational results for the case where $V_F = S \cup C$. In the first three columns of the tables, we provide instance data: n, K and an instance identifier (a, b, c, d, e). In the next seven columns, we report on results for PDPCD: the best lower (BLB) and upper (BUB) bounds obtained by BP at termination, the corresponding duality gap (%), the time (in seconds) needed by BP to solve the instance ($t(s)$), the number of nodes investigated in the search tree, followed by the number of pickup and delivery routes (PDP) and by the number of routes of type 1 in the best solution found for PDPCD. An entry "tl" at columns $t(s)$ indicates that the algorithm was unable to solve the instance within the imposed time limit of 14400 seconds. Similar entries are given next for BP#2 introduced on chapter 3 for VRPCD, in the columns under headings "BP#2 for VRPCD". According to the previous results for

instance			BP for PDPCD							BP#2 for VRPCD		
								# routes				
n	K	id	BLB	BUB	gap(%)	t(s)	nodes	PDP	type1	BLB	BUB	t(s)
		a	1647.7	1647.7	-	0.63	1	1	3	1659.9	1659.9	0.4
		b	2063.2	2063.2	-	707	3035	2	2	2130.1	2130.1	1525
10	4	c	1872.5	1872.5	-	0.54	1	1	3	1921.9	1921.9	94
		d	1740.5	1740.5	-	0.49	1	1	3	1776.9	1776.9	1865
		e	1947.1	1947.1	-	1009	3735	1	3	1950.5	1950.5	61
		a	2562.2	2562.2	-	0.9	1	2	4	2594.0	2594.0	1.2
		b	2894.0	2894.0	-	9836	12561	2	4	2911.0	2929.6	tl
15	6	c	2671.9	2693.6	0.81	tl	18047	1	5	2722.4	2742.3	tl
		d	2577.4	2577.4	-	208	289	4	2	2636.7	2679.6	tl
		e	2824.5	2832.0	0.26	tl	17981	1	5	2852.8	2852.8	13756
		a	3145.1	3182.2	1.17	tl	6592	3	4	3173.2	3202.9	tl
		b	3245.6	3248.5	0.09	tl	6415	4	3	3325.3	3383.7	tl
20	7	c	2951.3	2951.3	-	139	69	3	4	3067.0	3105.3	tl
		d	3264.8	3319.7	1.65	tl	7729	4	3	3311.0	3358.3	tl
		e	2991.4	3000.6	0.31	tl	3419	3	4	3014.8	3065.9	tl
		a	4031.4	4046.6	0.36	tl	1855	2	7	4055.9	4060.0	tl
		b	4149.8	4149.8	-	10495	2341	7	2	4243.7	4318.4	tl
25	9	c	3594.9	3594.9	-	16	1	4	5	3668.7	3728.7	tl
		d	3948.6	3959.4	0.28	tl	2991	5	4	3986.5	4003.3	tl
		e	4070.4	4143.2	1.76	tl	1392	2	7	4130.7	4191.8	tl
		a	4550.3	4550.3	-	14078	1221	4	6	4644.0	4664.6	tl
		b	4247.0	4290.4	1.01	tl	1170	8	2	4343.7	4394.7	tl
30	10	c	4652.2	4706.6	1.16	tl	556	10	0	4731.2	4771.3	tl
		d	4125.1	4226.2	2.38	tl	41	5	5	4184.7	4280.3	tl
		e	4514.7	4621.0	2.30	tl	461	5	5	4586.0	4695.2	tl

Table 4.1: Comparing Branch-and-price algorithms for PDPCD and VRPCD on solving instances of set 1: $\{c_i = 0 : i = 1, \ldots, n\}$

VRPCD, DP performs good on pricing routes for instances of set 1 (strictly constrained instances). For this reason, we set up DP algorithms for solving pricing problems for such experiments.

As it can be appreciated, the best lower bound provided for a VRPCD instance is almost always greater than the best upper bound available for the corresponding PDPCD instance of set 1. Considering only those instances solved to optimality by both methods (for all values of c_i tested here), no exception to that rule was observed. In such cases, the average cost reduction due to the introduction of pickup and delivery routes is 3.3%. In some cases ($c_i = 40, n = 20, K = 7$), optimal solutions to PDPCD are up to 7.1% cheaper than VRPCD counterparts. The fact that pickup and delivery routes indeed helps in reducing optimal costs is confirmed by the fact that, in all cases, at least one pickup and delivery route is found among the best known set of K routes for

instance			BP for PDPCD							BP#2 for VRPCD		
								# routes				
n	K	id	BLB	BUB	gap(%)	t(s)	nodes	PDP	type1	BLB	BUB	t(s)
		a	1703.9	1703.9	-	0.49	1	4	0	1749.7	1749.7	0.26
		b	2103.2	2103.2	-	4	29	2	2	2186.9	2186.9	98
10	4	c	1918.1	1918.1	-	1	5	4	0	1977.5	1977.5	0.65
		d	1804.1	1804.1	-	0.39	1	2	2	1843.5	1843.5	20
		e	2027.1	2027.1	-	56	227	3	1	2044.8	2044.8	10
		a	2664.8	2664.8	-	21	39	2	4	2714.0	2714.0	0.62
		b	2955.1	2955.1	-	10	19	2	4	3013.8	3013.8	3
15	6	c	2752.6	2752.6	-	133	209	3	3	2868.4	2875.5	tl
		d	2617.4	2617.4	-	3	1	4	2	2756.2	2756.2	678
		e	2922.5	2922.5	-	15	29	3	3	2970.2	2970.2	984
		a	3256.4	3256.4	-	128	53	7	0	3341.9	3386.9	tl
		b	3325.9	3325.9	-	3641	1715	4	3	3476.7	3476.7	8289
20	7	c	3019.2	3019.2	-	6	3	5	2	3211.9	3211.9	674
		d	3375.0	3375.0	-	5007	2253	4	3	3485.9	3572.9	tl
		e	3084.9	3084.9	-	115	23	4	3	3170.8	3170.8	3277
		a	4135.2	4135.2	-	47	3	9	0	4240.0	4251.0	tl
		b	4167.1	4167.1	-	16	1	9	0	4445.9	4539.8	tl
25	9	c	3691.7	3691.7	-	283	25	6	3	3860.7	3860.7	3857
		d	4000.1	4000.1	-	8	1	7	2	4175.5	4175.5	7791
		e	4191.5	4248.0	1.33	tl	1209	5	4	4295.7	4377.8	tl
		a	4635.7	4635.7	-	32	1	10	0	4789.4	4794.0	tl
		b	4317.1	4327.6	0.24	tl	903	8	2	4526.7	4607.5	tl
30	10	c	4706.6	4706.6	-	4001	83	10	0	4964.3	4964.3	7456
		d	4228.8	4246.0	0.4	tl	42	8	2	4396.3	4454.7	tl
		e	4606.5	4630.9	0.53	tl	533	10	0	4752.1	5021.9	tl

Table 4.2: Comparing Branch-and-price algorithms for PDPCD and VRPCD on solving instances of set 1: $\{c_i = 20 : i = 1, \ldots, n\}$

PDPCD. It should be pointed out that, as the loading/unloading costs increase, fewer routes of type 1 (more pickup and delivery routes) are included in the best solutions for PDPCD (see column # of routes in Table 4.3). Therefore, irrespective of how arc costs compare to load changing costs, PDPCD solutions seem to benefit from routes that allow vehicles to avoid the stop at CD.

From a computational point of view, BP for PDPCD seems to work better than the BP#2 algorithm for VRPCD. Out of the 75 cases considered for set 1, BP#2 managed to solve 38 instances to proven optimality, while 54 of them were solved within the time limit by BP for PDPCD. Considering those instances left unsolved, duality gaps attained by BP for PDPCD and BP#2 for VRPCD are around 0.84% and 1.21%, respectively.

Let us now discuss how BP for PDPCD performs on solving instances of set 2.

instance			BP for PDPCD							BP#2 for VRPCD		
								# routes				
n	K	id	BLB	BUB	gap(%)	t(s)	nodes	PDP	type1	BLB	BUB	t(s)
10	4	a	1703.9	1703.9	-	0.51	1	4	0	1809.8	1809.8	0.14
		b	2119.0	2119.0	-	12	65	4	0	2226.9	2226.9	2
		c	1918.1	1918.1	-	0.18	5	4	0	2017.5	2017.5	0.21
		d	1824.1	1824.1	-	0.51	1	2	2	1890.4	1890.4	4
		e	2032.0	2032.0	-	0.29	1	4	0	2124.0	2124.0	21
15	6	a	2694.4	2694.4	-	1	1	6	0	2802.2	2802.2	0.45
		b	2975.6	2975.6	-	3	5	6	0	3073.8	3073.8	0.26
		c	2759.4	2759.4	-	76	89	6	0	2938.5	2938.5	573
		d	2639.7	2639.7	-	14	13	6	0	2796.2	2796.2	42
		e	2951.5	2951.5	-	1	1	4	2	3027.9	3027.9	20
20	7	a	3256.4	3256.4	-	5	1	7	0	3430.3	3454.4	tl
		b	3327.7	3327.7	-	414	193	7	0	3526.5	3526.5	549
		c	3059.2	3059.2	-	183	71	5	2	3295.3	3295.3	248
		d	3390.6	3390.6	-	181	79	7	0	3573.9	3576.5	tl
		e	3087.9	3087.9	-	91	19	7	0	3237.1	3237.1	18
25	9	a	4135.2	4135.2	-	64	3	9	0	4334.9	4334.9	14208
		b	4167.1	4167.1	-	14	1	9	0	4515.1	4521.5	tl
		c	3713.5	3729.2	0.42	tl	1369	9	0	3920.7	3920.7	963
		d	4020.1	4020.1	-	11	1	9	0	4204.0	4204.0	21
		e	4242.9	4284.0	0.96	tl	971	9	0	4375.9	4462.0	tl
30	10	a	4635.7	4635.7	-	41	1	10	0	4860.2	4864.1	tl
		b	4345.6	4345.6	-	3417	161	10	0	4580.5	4619.2	tl
		c	4706.7	4706.7	-	1332	27	10	0	5038.6	5045.0	tl
		d	4247.1	4253.5	0.15	tl	33	8	2	4478.2	4532.3	tl
		e	4610.7	4630.9	0.44	tl	418	10	0	4847.1	4961.2	tl

Table 4.3: Comparing Branch-and-price algorithms for PDPCD and VRPCD on solving instances of set 1: $\{c_i = 40 : i = 1, \ldots, n\}$

According to previous experiments for VRPCD on chapter 3, DP algorithms performs poorly on solving pricing problems of loose constrained instances. For that reason, we replaced DP algorithms by a combination of GRASP + BC algorithms to solve both pricing problems: *docking routes* and *pickup and delivery routes*. In such a way, BP for PDPCD does not make we use of DP algorithm on solving pricing problems for instances of set 2, only a combination of BC + GRASP. We illustrate in table 4.4 the results of BP for PDPCD for solving instances of set 2 with loading/unloading costs $c_i = 20$. Entries on such table are similar from that on tables 4.1-4.3 for set 1. However, we used here BP#1 to evaluate VRPCD solutions, cause it performs better than BP#2 on solving instances of set 2. In addition, because BP for PDPCD is not capable of evaluating the LP bounds for some instances of set 2, we used (as in section 3) a Lagrangian Relaxation lower bound, which is indicated on tables with a symbol $*$.

instance			BP for PDPCD							BP#1 for VRPCD		
								# routes				
n	K	id	BLB	BUB	gap(%)	t(s)	nodes	PDP	type1	BLB	BUB	t(s)
		a	1073.9	1073.9	-	497	173	0	2	1073.9	1073.9	26
		b	1318.8	1318.8	-	2444	485	2	0	1344.3	1344.3	34
10	2	c	1232.3	1232.3	-	13	3	2	0	1288.5	1288.5	29
		d	1087.4	1087.4	-	11	1	2	0	1159.6	1159.6	41
		e	1337.6	1337.6	-	313	113	2	0	1357.7	1357.7	25
		a	1616.6	1628.0	0.7	tl	481	3	0	1633.2	1633.2	359
		b	1724.4	1724.4	-	289	9	0	3	1724.4	1724.4	498
15	3	c	1652.6	1652.6	-	1781	107	3	0	1664.9	1664.9	454
		d	1581.0	1581.0	-	100	5	3	0	1636.9	1636.9	502
		e	1754.2	1770.9	0.9	tl	361	0	3	1767.2	1767.2	396
		a	1985.7	1985.7	-	1015	1	4	0	2022.4	2196.8	tl
		b	2110.2	2154.1	2.0	tl	45	4	0	2066.6	2213.3	tl
20	4	c	1916.0	1916.0	-	4011	19	4	0	1988.5	2135.1	tl
		d	2082.2	2082.2	-	3726	19	4	0	2097.8	2147.4	tl
		e	1972.0	1972.0	-	603	3	4	0	1987.9	2196.5	tl
		a	2483.9	4708.9	47.2	tl	12	0	5	2400.6	2686.0	tl
		b	2521.1	2591.6	2.7	tl	19	5	0	2524.7	2784.5	tl
25	5	c	2243.6	4464.5	49.7	tl	11	0	5	2183.3	2393.0	tl
		d	2433.7	2485.0	2.0	tl	8	5	0	2413.5	2663.2	tl
		e	2603.2	4634.8	43.8	tl	14	0	5	2501.4	2759.7	tl
		a	2239.6_*	2779.1	19.4	tl	1	5	0	2570.7	5250.0	tl
		b	2302.7_*	2449.7	6.0	tl	1	5	0	2349.3	2786.0	tl
30	5	c	2600.7_*	2697.6	3.5	tl	1	5	0	2611.5	2880.4	tl
		d	2435.0_*	4256.9	42.8	tl	1	0	5	2378.2	2856.7	tl
		e	2520.8_*	5684.3	55.6	tl	1	0	5	2643.4	3261.2	tl

Table 4.4: Comparing Branch-and-price algorithms for PDPCD and VRPCD on solving instances of set 2: $\{c_i = 20 : i = 1, \ldots, n\}$

Because instances of set 2 are harder to evaluate, BP for PDPCD was capable to provide optimality certificates for only 12 instances. From 10 of such instances the optimal costs provided by PDPCD improve on results for VRPCD. However, differently from instances of set 1, where optimal solutions involve both types of routes, here only *docking routes* or *pickup and delivery routes* are used to solve each instance. We believe that such a behavior is due to the reduced number of vehicle of instances of set 2, but it is not clear for us.

Our last experiment is proposed to investigate how BP for PDPCD works for the case where $V_F \neq S \cup C$. For that, we have partitioned S according to: $S_{CD} := \left\{i \in S : 1 \leq i \leq \left\lfloor \frac{n}{3} \right\rfloor\right\}$, $S_{NS} := \left\{i \in S : \left\lfloor \frac{n}{3} + 1 \right\rfloor \leq i \leq \left\lfloor \frac{2n}{3} \right\rfloor\right\}$ and $S_F = \left\{i \in S : \left\lfloor \frac{2n}{3} + 1 \right\rfloor \leq i \leq n\right\}$. In Table 4.5, we indicate results for instances of set 1 considering loading/unloading costs $c_i = 40$. Columns in Table 4.5 have the same meaning

as the first ten columns in Tables 4.1-4.4, described earlier.

instance								# routes	
n	K	id	BLB	BUB	gap(%)	t(s)	nodes	PDP	type1
		a	1939.4	1939.4	-	0.04	3	3	1
		b	2214.3	2214.3	-	2	267	2	2
10	4	c	2094.2	2094.2	-	0.06	3	3	1
		d	1911.5	1911.5	-	0.08	5	2	2
		e	2224.6	2224.6	-	0.03	1	2	2
		a	2851.9	2851.9	-	18	1343	2	4
		b	3031.2	3031.2	-	0.05	1	3	3
15	6	c	2848.4	2848.4	-	0.16	1	3	3
		d	2770.3	2770.3	-	6	145	3	3
		e	3079.3	3079.3	-	0.25	13	3	3
		a	3382.8	3382.8	-	16	149	5	2
		b	3471.9	3471.9	-	31	253	4	3
20	7	c	3323.0	3323.0	-	6899	23037	3	4
		d	3585.5	3585.5	-	76	467	4	3
		e	3215.1	3215.1	-	255	395	4	3
		a	4360.5	4390.3	0.67	tl	5729	4	5
		b	4446.7	4450.7	0.09	tl	9638	4	5
25	9	c	3871.1	3871.1	-	501	373	5	4
		d	4196.6	4196.6	-	2	1	4	5
		e	4350.4	4350.4	-	223	77	5	4
		a	4870.7	4870.7	-	16	40	5	5
		b	4572.3	4572.3	-	13504	8355	4	6
30	10	c	4950.9	4950.9	-	6947	1894	6	4
		d	4478.2	4478.2	-	3615	168	5	5
		e	4897.7	4927.9	0.61	tl	5014	6	4

Table 4.5: PDPCD - $V_F \neq S \cup C$ - $\{c_i = 40 : i = 1, \ldots, n\}$

Results in Table 4.5 indicate that, compared to the case where $V_F = S \cup C$, similar sets of instances are solved by BP. However, BP usually solves the instances with less computational effort. For those instances left unsolved, smaller duality gaps are obtained at the end of the time limit, since each node in the tree takes much less computational effort to be evaluated. That happens since the graphs involved in the pricing problems for $r \in R$ and $r \in R_d$ usually have fewer nodes, (many nodes are forbidden for each pricing problem) and, thus, DP is likely to work faster.

4.5 Concluding Remarks

As an attempt to better integrate routing and scheduling decisions in cross-docking systems, we introduced the Pickup and Delivery Problem with Cross Docking (PDPCD).

Compared to the Vehicle Routing Problem with Cross-Docking (VRPCD), the model proposed here allows routes that do not stop at the cross-docking to be chosen. In doing so, our aim was to reduce total distribution costs, since imposing a stop at the cross-docking may not be cost effective. For solving PDPCD, a Branch-and-price algorithm was implemented.

Our computational results confirmed the claim that allowing the new type of routes would reduce total transportation costs. Optimal solutions for PDPCD are, on the average, at least 3.7% cheaper than optimal solutions for VRPCD. Sometimes they are 7.1% cheaper than VRPCD counterparts. As a by product, the Branch-and-price algorithm implemented for PDPCD works better than similar approaches for VRPCD, since *pickup and delivery routes* can be priced out faster than *docking routes* due to precedence constraints, which reduces the number of solutions to be investigated by pricing algorithms.

Chapter 5

A Branch-and-cut-and-price algorithm for the Two-Echelon Capacitated Vehicle Routing Problem

We introduce in this chapter an exact solution approach for 2E-CVRP which improves on our previous work [Santos et al., 2012a]. We propose a new IP set partitioning reformulation for 2E-CVRP that, differently from the formulation in Santos et al. [2012a], does not involve variables indexed by vehicles and, thus, does not suffer from symmetry issues. We also combine, in a single algorithm, column and row generation. On the one hand, the algorithm is based on a reformulation that relaxes route elementarity and prices q-routes. On the other hand, it separates several classes of valid inequalities within a Branch-and-cut-and-price (BCP) framework. In doing so, pricing problems are solved much faster than what would otherwise be if routes satisfying the elementarity conditions were explicitly priced out. Due to the dynamic generation of cutting planes, we close a significant part of the additional duality gap that is implied, when route elementarity is relaxed.

The 2E-CVRP is formally defined in Section 5.1, where we also discuss the valid inequalities behind our BCP algorithm. Details on the BCP implementation are given next, in Section 5.2. In Section 5.3, we report on our computational experiments and compare BCP results to existing 2E-CVRP exact previous approaches [Santos et al., 2012a; Perboli et al., 2011; Jepsen et al., 2012]. Finally, we close the the paper in Section 5.4, where we offer some conclusions.

5.1 Integer Programming Reformulations

Assume that $G = (V, A)$ denotes a directed graph with set of vertices $V := \{\{0\}, S, C\}$, where 0 is a vertex that represents the depot and $S := \{s_1, \ldots, s_k\}$ and $C := \{c_1, \ldots, c_n\}$ respectively denote sets of satellites and customers. Let the arc set A be partitioned into two disjoint subsets A_1 and A_2, i.e., $A := A_1 \cup A_2$ ($A_1 \cap A_2 = \emptyset$). Arc set $A_1 := \{(i, j) : i, j \in \{0\} \cup S\}$ denotes the set of level-1 arcs, which connect the satellites and the depot. Likewise, $A_2 := \{(i, j) : i, j \in C\} \cup \{(i, j) : i \in S, j \in C \text{ or } i \in C, j \in S\}$ denotes the set of level-2 arcs, which connect customers and satellites.

Let $\mathcal{K}_1$ denote a fleet of $K_1 = |\mathcal{K}_1|$ homogeneous vehicles, available to implement level-1 routes. Each vehicle in this fleet has the same capacity Q_1. Likewise, to implement level-2 routes, another fleet $\mathcal{K}_2$ of $K_2 = |\mathcal{K}_2|$ homogeneous vehicles (each one with capacity Q_2) is also available. Whenever a vehicle traverses $(i, j) \in A_1$, a transportation cost $c^1_{ij} \geq 0$ is incurred. Similarly, a cost $c^2_{ij} \geq 0$ is to be paid, whenever $(i, j) \in A_2$ is traversed by a level-2 vehicle. Arc costs in both levels do satisfy the triangle inequalities.

It is assumed that demands of the same good are assigned to all customers. To each customer $i \in C$, an amount $d_i > 0$ of the same good must be delivered. These demands are available at the central depot, vertex 0. 2E-CVRP consists in finding up to K_1 level-1 routes, from the depot to the satellites and up to K_2 level-2 routes from the satellites to the customers. The total number of vehicles available in each level may not be used.

One important distinction is made on the number of satellites allowed to be visited by each level-1 and level-2 route. While in the first case each route must start at the depot and may visit one or more satellites, level-2 routes cannot span more than one satellite. Instead, each level-2 route must start at a given satellite, visit some customers precisely once and then return to the same satellite. In order to avoid congestion in the satellites, at most m_s ($m_s \geq 1$) level-2 routes are allowed to leave each satellite $s \in S$. Every customer, however, must be visited exactly once.

Differently from other routing problems in the literature, 2E-CVRP allows a satellite (which plays the role of a customer in level-1 routes) to be visited by more than one level-1 route. If that were not the case, the total demand shipped from the depot to each satellite would be bounded by Q_1. On the one hand, cheaper solutions can be obtained when the total amount of goods supplied to each satellite is greater than the level-1 capacity. On the other hand, for some instances, the total demand $\sum_{i \in C} d_i$ would eventually exceed $|S|Q_1$ and the instance would be unfeasible.

Whenever goods are loaded/unloaded at a satellite $s \in S$, a cost L_s (per unit of

good) is incurred. The goal in 2E-CVRP consists in finding level-1 and level-2 routes, in order to minimize the total cost, given by the sum of the traveling costs in both levels plus the cost of loading/unloading operations at the satellites.

The 2E-CVRP definition provided above, in terms of its objective function and constraints, follows that introduced in Perboli et al. [2011] and later considered in Santos et al. [2012a]. Different multi-echelon problems aiming to optimize other costs functions and including additional constraints could be found, for example, in Dondo et al. [2009, 2011]; Topan et al. [2009].

5.1.1 A reformulation that relies on routes that satisfy the elementarity condition

In order to formulate 2E-CVRP as an Integer Program (IP), consider the following notation and definitions. Let R_1 denote the set of all level-1 routes. Each route $r \in R_1$ starts at the depot, visits a set of satellites and return to depot. Likewise, for a given satellite $s \in S$, let R_2^s denote the set of all level-2 routes that start and end at s. For each $r \in \bigcup_{s \in S} R_2^s \cup R_1$, let c_r denote its cost (given by the sum of the costs of its arcs). Denote by a_{ir} a binary parameter that indicates whether route r visits ($a_{ir} = 1$) or not ($a_{ir} = 0$) vertex $i \in V$. It is assumed that each route $r \in \bigcup_{s \in S} R_2^s$ does not violate capacity Q_2, i.e., $\sum_{i \in C} a_{ir_2} d_i \leq Q_2, r_2 \in R_2$.

The 2E-CVRP IP reformulation given here makes use of three sets of decision variables:

- non-negative integer valued variables λ_{r_1}, to indicate how many vehicles implement route $r_1 \in R_1$;
- binary variables $\gamma_{r_2}^s$, to indicate whether ($\gamma_{r_2}^s = 1$) or not ($\gamma_{r_2}^s = 0$) a route $r_2 \in R_2^s$ is implemented;
- non-negative real valued variables δ^{sr_1}, to indicate the amount of goods delivered from the depot to satellite $s \in S$ using route $r_1 \in R_1$. Although variables δ^{sr_1} are not integer constrained, they assume integer values as long as variables λ_{r_1} and $\gamma_{r_2}^s$ and problem data (Q_1, Q_2 and $\{d_i : i \in C\}$) are integers.

An IP reformulation for 2E-CVRP is implied by:

$$\min \sum_{r_1 \in R_1} c_{r_1} \lambda_{r_1} + \sum_{s \in S} \sum_{r_2 \in R_2^s} c_{r_2} \gamma_{r_2}^s + \sum_{s \in S} \sum_{r_1 \in R_1} L_s \delta^{sr_1} \tag{5.1}$$

$$\sum_{r_1 \in R_1} \lambda_{r_1} \leq K_1 \tag{5.2}$$

$$\sum_{s \in S} \sum_{r_2 \in R_2^s} \gamma_{r_2}^s \leq K_2 \tag{5.3}$$

$$\sum_{r_2 \in R_2^s} \gamma_{r_2}^s \leq m_s \qquad \forall s \in S \tag{5.4}$$

$$\sum_{s \in S} \sum_{r_2 \in R_2^s} a_{ir_2} \gamma_{r_2}^s = 1 \qquad \forall i \in C \tag{5.5}$$

$$\sum_{r_2 \in R_2^s} \gamma_{r_2}^s \sum_{i \in C} a_{ir_2} d_i = \sum_{r_1 \in R_1} \delta^{sr_1} \qquad \forall s \in S \tag{5.6}$$

$$\sum_{s \in S} a_{sr_1} \delta^{sr_1} \leq Q_1 \lambda_{r_1} \qquad \forall r_1 \in R_1 \tag{5.7}$$

$$\sum_{r_1 \in R_1} \sum_{s \in S} (1 - a_{sr_1}) \delta^{sr_1} = 0 \tag{5.8}$$

$$\lambda_{r_1} \in \mathbb{N} \qquad \forall r_1 \in R_1 \tag{5.9}$$

$$\gamma_{r_2}^s \in \mathbb{B} \qquad \forall s \in S,\ \forall r_2 \in R_2^s \tag{5.10}$$

$$\delta^{sr_1} \in \mathbb{R}_+ \qquad \forall s \in S,\ \forall r_1 \in R_1 \tag{5.11}$$

Constraints (5.2) and (5.3) enforce that no more than K_1 and K_2 vehicles are used to respectively implement level-1 and level-2 routes. Constraints (5.4) impose that no more than m_s level-2 vehicles leave satellite $s \in S$. Set partitioning constraints (5.5) guarantee that each customer is visited exactly once. Constraints (5.6) couple δ^{sr_1} and $\gamma_{r_2}^s$ variables. They state flow balance conditions at the satellite, i.e., the amount of goods delivered to the customers by one satellite is precisely what is supplied to that satellite from the depot, by level-1 routes. Constraints (5.7) couple δ^{sr_1} and λ_{r_1} variables and guarantee that the capacity of a level-1 vehicle is never exceeded. Constraint (5.8), on the other hand, is used to ensure that whenever $a_{sr_1} = 0$ so does the corresponding variable δ^{sr_1}.

Since the total amount of goods supplied to each satellite is not known beforehand, we could not state level-1 capacity constraints in the same way we did for level-2 routes. In other words, we could not guarantee that the capacity Q_1 would not be exceeded by simply restricting the sets of satellites visited by each route $r \in R_1$. Therefore, in our model, level-1 capacity constraints are explicitly considered in the master program, due to the inclusion of inequalities (5.7).

Once there is one constraint (5.7) for each $r_1 \in R_1$, the previous observation might suggest that rows (as well as columns) associated to routes $r_1 \in R_1$ would be dynamically generated in our algorithm for 2E-CVRP. That is not the case. Since

the number of satellites is usually small (bounded by 5 in all instances of our test set) compared to the number of customers , we can explicitly handle all variables associated to level-1 routes (δ^{sr_1} and λ_{r_1}) in the model. Likewise, all constraints (5.7) are going to be explicitly kept in the model. This way, only $\gamma^s_{r_2}$ variables will be dynamically generated by a Column Generation (CG) approach.

It is noteworthy that formulation (5.1)-(5.11) is quite similar to that introduced in our previous study [Santos et al., 2012a]. However, they differ in a very important aspect. That formulation makes use of binary decision variables $\lambda^k_{r_1}$ to define whether or not a given vehicle $k \in \mathcal{K}_1$ implements route $r_1 \in R_1$. Since vehicles are identical, that formulation suffers from symmetry issues. Aiming to overcome them, we now replace previous binary variables $\{\lambda^k_{r_1} : k = 1, \ldots, K_1\}$ by an aggregated integer variable λ_{r_1} and change coupling constraints accordingly. Note that according to the definitions, old and new variables relate as $\lambda_{r_1} = \sum_{k \in \mathcal{K}_1} \lambda^k_{r_1}$ forall $r_1 \in R_1$. As a result of stronger coupling constraints (5.6) and (5.7), our new formulation provides bounds that are stronger than those implied by the model in Santos et al. [2012a].

The LP bounds provided by model (5.1)-(5.11) can be strengthened by the following valid inequality for 2E-CVRP:

$$\sum_{r_1 \in R_1} \lambda_{r_1} \geq \left\lceil \frac{\sum_{i \in C} d_i}{Q_1} \right\rceil . \tag{5.12}$$

Such a constraint imposes a minimum number of level-1 vehicles required to be in duty, so that the total demand can be shipped from the depot to the satellites.

From now on, let F1 denote formulation (5.1)-(5.12). Denote by LP(F1) the Linear Programming (LP) relaxation lower bound implied by F1, i.e., LP(F1) is the optimal value of the LP implied by (5.1)-(5.12), when constraints (5.9) and (5.10) are respectively replaced by their continuous counterparts: $\{\lambda_{r_1} \geq 0 : r_1 \in R_1\}$ and $\{\gamma^s_{r_2} \geq 0 : s \in S,\ r_2 \in R_2\}$. Note that, since variables $\gamma^s_{r_2}$ are non-negative and due to (5.5), constraints $\{\gamma^s_{r_2} \leq 1 : s \in S,\ r_2 \in R^s_2\}$ do not need to be explicitly imposed for the definition of the LP relaxation of F1.

The evaluation of LP(F1) suggests the use of a CG algorithm, since F1 includes exponentially many variables $\gamma^s_{r_2}$. Recall that decision variables $\gamma^s_{r_2}$ are associated to level-2 routes, i.e., routes that satisfy the elementarity condition. Therefore, pricing a level-2 route consists in solving an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) [Feillet et al., 2004; Ropke and Cordeau, 2006]. Note that, the dual variables associated to constraints (5.3)-(5.4) in the LP relaxation of F1 may assume negative values. Thus, the graph over which ESPPRC is formulated and solved

may involve cycles of negative cost. As a result, ESPPRC is NP-Hard [Feillet et al., 2004].

5.1.2 A reformulation based on q-routes

Evaluating LP(F1) bounds is very time consuming. One alternative to obtain lower bounds for 2E-CVRP in an expedite fashion consists in allowing level-2 routes to visit some vertices more than once, i.e., relaxing level-2 routes' elementarity conditions. In one of such possible relaxations, level-2 routes still satisfy the capacity constraints, but a given customer may be visited more than once, in the same route. Such routes are called q-routes. Their pricing problem can be formulated as a Shortest Path Problem with Capacity Constraints (SPPCC), for which a pseudo-polynomial time algorithm does exist [Christofides et al., 1981].

This relaxation gives rise to another refomulation for 2E-CVRP, where decision variables $\gamma^s_{r_2}$ are assigned to q-routes. In order to state the second reformulation, assume now that $\mathcal{R}^s_2$ denotes the set of all level-2 q-routes. Note that, according to our definitions, $R^s_2 \subset \mathcal{R}^s_2$. While for $r \in R_2$, $a_{ir_2} \in \{0, 1\}$, for q-routes that do not satisfy the elementarity condition, parameter a_{ir} (which denotes the number of times vertex $i \in C$ visited by route $r_2 \in \mathcal{R}^s_2$) may assume integer values greater than one.

From now on, assume that F2 denotes the second reformulation, implied by (5.1)-(5.12), after assinging $\gamma^s_{r_2}$ variables to q-routes and redefining a_{ir_2} accordingly. Likewise, let LP(F2) denote the implied LP relaxation lower bound.

In our previous work [Santos et al., 2012a], we implemented Branch-and-price algorithms based on formulations that, like F1 and F2, considered routes that satisfy elementartity conditions and q-routes, respectively. F1 is actually stronger than the formulation in Santos et al. [2012a] that makes use of routes that satisfy elementarity conditions. That applies since we now make use of different (not indexed by k) level-2 variables and obtain tighter coupling constraints. By similar arguments, F2 (which is derived from F1) is also stronger than its counterpart in Santos et al. [2012a] that makes use of q-routes.

The BCP algorithm implemented here actually does not rely on formulation F1 to generate 2E-CVRP lower bounds. Instead, it makes use of formulation F2, strengthened with several classes of valid inequalities. In doing so, the goal is to obtain 2E-CVRP lower bounds close to LP(F1), with much less computational effort. Therefore, the algorithm introduced here prices q-routes and separates valid inequalities in rounds of column and row generation. Inequalities separated by our algorithm are described next.

5.1.3 Valid Inequalities

We decided to separate only inequalities involving level-2 variables. Such a decision was motivated by the fact that level-1 routes satisfy the elementarity conditions and the impact of the inclusion of additional cuts involving level-1 variables in the lower bounds should be marginal.

To simplify the presentation, the inequalities are not stated in terms of variables $\gamma^s_{r_2}$. Instead, we make use of binary decision variables $\{x^s_{ij} : (i,j) \in A_2, s \in S\}$ to indicate whether ($x^s_{ij} = 1$) or not ($x^s_{ij} = 0$) an arc $(i,j) \in A_2$ is included in a q-route that starts and ends at s. In order to relate these variables with $\gamma^s_{r_2}$, let us make use of an integer parameter $b_{ijr_2} : (i,j) \in A_2, r_2 \in \cup_{s\in S}\mathcal{R}^s_2$ that indicates how many times an arc (i,j) is traversed by a q-route $r_2 \in \mathcal{R}^s_2$. We also use binary variables y^s_i to represent whether or not a vertex $i \in C$ is visited by a q-route $r_2 \in \mathcal{R}^s_2$. According to these definitions, $x^s_{ij} = \sum_{r_2\in\mathcal{R}^s_2} b_{ijr_2}\gamma^s_{r_2}$, $x_{ij} = \sum_{s\in S} x^s_{ij}$ and $y^s_i = \sum\limits_{r_2\in\mathcal{R}^s_2} a_{ir_2}\gamma^s_{r_2}$. In doing so, any 2E-CVRP valid inequality written in terms of x_{ij}, x^s_{ij} and y^s_i can be conveniently re-written in terms of $\gamma^s_{r_2}$ variables.

Inequalities separated by our algorithm are borrowed from the TSP and CVRP literature: rounded capacity inequalities, strengthened combs, homogeneous multistar and subtour breaking constraints. The first three families are described in terms of aggregated variables x_{ij} and their separation is conducted over the support graph implied by the aggregated variables. Subtour breaking constraints, on the other hand, are stated for decision variables involving one satellite at a time. Accordingly, their separation makes use of a support graph associated to variables involving only one satellite.

To be more precise, assume that at a given LP relaxation, $\hat{\mathcal{R}}^s_2$ denotes the set of q-routes that are included in the current restricted linear programming master problem. Define $\hat{x}^s_{ij} = \sum_{r_2\in\hat{\mathcal{R}}_2} b_{ijr_2}\gamma^s_{r_2}$, $\hat{x}_{ij} = \sum_{s\in S}\hat{x}^s_{ij}$ and $\hat{y}^s_i = \sum_{r_2\in\hat{\mathcal{R}}_2} a_{ir_2}\gamma^s_{r_2}$. Let $\hat{G} = (\hat{V}, \hat{A})$, where $\hat{V} = C \cup \hat{0}$ where $\hat{0} := \bigcup_{i\in S} s_i$ denotes an aggregated depot and $\hat{A} = \{(i,j) \in A_2 : \hat{x}_{ij} > 0\}$. Likewise, let $\hat{G}^s = (\hat{V}^s, \hat{A}^s)$, where $\hat{V}^s := \{i \in C : \hat{y}^s_i > 0\}$ and $\hat{A}^s := \{(i,j) \in A_2 : \hat{x}^s_{ij} > 0\}$.

5.1.3.1 Rounded Capacity

Inequalities

$$\sum_{i\in T}\sum_{j\in C\setminus T}(x_{ij} + x_{ji}) \geq 2\pi(T), \qquad \forall T \subset C : |T| \geq 1 \tag{5.13}$$

are valid for CVRP [Fukasawa et al., 2006]. In such an inequality, $\pi(T)$ represents the minimum number of vehicles that must serve a given set $T \subseteq C$. Finding $\pi(T)$ amounts in solving a Bin Packing Problem, and thus, the separation of (5.13) is NP-Hard. Inequalities weaker than (5.13) can be obtained if $\pi(T)$ is replaced by a valid lower bound $\pi_{LB}(T)$ on $\pi(T)$. Depending on which lower bounds are used, different versions of the capacity constraints are obtained [Araque et al., 1990; Campos et al., 1991; Cornuéjols and Harche, 1993; Naddef and Rinaldi, 2002; Lysgaard et al., 2004].

In this paper, we separate the rounded capacity inequalities, which are obtained by setting $\pi_{LB}(T) = \left\lceil \frac{\sum_{i \in T} d_i}{Q_2} \right\rceil$. They represent a good compromise between lower bound improvements and the effort involved in their separation, which is also NP-Hard [Naddef and Rinaldi, 2002]. To separate them, we resort to the heuristic in Lysgaard et al. [2004]. It first tries to find violated cuts for sets of vertices T implied by the vertices in the same connected components of $\hat{G}$. If no violated inequalities are found, then it uses shrinking techniques on the supporting graph aiming to investigate other promising sets T, until no more sets may be investigated. The separation is conducted by the implementation available in the CVRPSEP package [Lysgaard, 2004].

5.1.3.2 Strengthened Combs

Comb inequalities were first recognized as $\{0, \frac{1}{2}\}$ Chvátal-Gomory cuts [Chvátal, 1973]. They were first used to strengthen relaxations for the TSP by Grötschel and Padberg [1979a,b]. Later, Laporte and Bourjolly [1984]; Laporte and Nobert [1984] extended them for the CVRP case. In its original CVRP form, Comb inequalities do not take into account the capacity of the vertices involved in their definition. That was only considered in the subsequent studies of Lysgaard et al. [2004].

In order to state them, define a subset $H \subset \hat{V}$, called *handle* and t subsets $T_1, \ldots, T_t \subset \hat{V}$, named *teeth*, such that:

- $H \cap T_j \neq \emptyset$ and $T_j \setminus H \neq \emptyset, \quad j = \{1, \ldots, t\}$;
- for each pair $\{i, j\} : i, j \in \{1, \ldots, t\}$ such that either $T_i \cap T_j \cap H = \emptyset$ or else $T_i \cap T_j \subset H$
- $t \geq 3$.

Consider $\pi(T)$ as defined before. In addition, let $\tilde{\pi}(T) = \pi(T)$ if $\hat{0} \in T$, or $\pi(\hat{V} \setminus T)$ if $\hat{0} \notin T$. Define $S(H, T_1, \ldots, T_j) := \sum_{j=1}^{t} (\tilde{\pi}(T_j \cap H) + \tilde{\pi}(T_j \setminus H) + \tilde{\pi}(T_j))$. Strengthened

Combs inequalities (SC) are stated as:

$$\sum_{i\in H}\sum_{j\in C\setminus H}(x_{ij}+x_{ji})+\sum_{j=1}^{t}\sum_{i\in T_j}\sum_{j\in C\setminus T_j}(x_{ij}+x_{ji})\geq S(H,T_1,\ldots,T_j)+1. \tag{5.14}$$

Their separation, in either their original or strengthened forms, is NP-Hard [Padberg and Rinaldi, 1990; Naddef and Thienel, 2002]. To separate them, we use a heuristic also proposed by [Lysgaard et al., 2004], whose implementation is available at the CVRP package [Lysgaard, 2004]. The heuristic replaces each term $\pi(\cdot)$ in the definition of $S(H,T_1,\ldots,T_j)$ by its relaxation $\pi_{LB}(\cdot)$, as done in the course of the separation procedure for rounded capacity cuts. Then, it shrinks vertices of graph looking for for *handle* and *teeth* candidates. If no violated inequality (5.14) is found in the resulting graph, it looks for one of its special cases, namely the 2-matching combs (where the right hand side in (5.14) is replaced by $3t+1$, and $|T_j|=2,\ \ j=\{1,\ldots,t\}$). That is accomplished by computing the Gomory-Hu tree by means of the polynomial-time exact algorithm of Padberg and Rao [1982].

5.1.3.3 Homogeneous Multistar

Multistar inequalities were introduced by Araque et al. [1990] for the CVRP with unit demands. Over the years, different authors generalize them for the CVRP case with general demands [Laporte, 1991; Fisher, 1994; Gouveia, 1995]. Homogeneous multistar inequalities [Letchford et al., 2002], which are the form used here, are defined in terms of three sets: $N\subset C$, $L\subset N$ and $T\subset C\setminus N$. They also involve three integer coefficients β,τ and σ, which depend on $|N|,|L|$ and $|T|$. Homogeneous Multistar (HM) inequalities are stated as:

$$\beta\sum_{i\in N}\sum_{j\in N}x_{ij}-\tau\sum_{i\in L}\sum_{j\in T}(x_{ij}+x_{ji})\leq\sigma. \tag{5.15}$$

In our study, we follow the approach proposed by Letchford et al. [2002] to separate HM. It is a greedy heuristic that searches for sets N,L, and T, according to the values $\{\hat{x}_{ij}:(i,j)\in\hat{A}\}$, and then defines β,τ and σ. The separation routines we actually used are, as in the case of RC and SC, those available for download at the CVRPSEP package [Lysgaard, 2004].

5.1.3.4 Generalized Subtour Breaking

Since q-routes may not satisfy elementarity, generalized subtour breaking constraints [Dantzig et al., 1954; Lucena, 1992; Margot et al., 1994; Goemans, 1994] may not be satisfied by the LP relaxations of LP(F2). For the 2E-CVRP case, these inequalities are stated as:

$$\sum_{i \in T} \sum_{j \in T} x_{ij}^s \leq \sum_{i \in T} y_i^s - 1 \qquad \forall s \in S,\ \forall T \subset C : |T| \geq 2. \tag{5.16}$$

Although for a given $s \in S$, their exact separation can be conducted in $O(|\hat{V}^s|^3)$ time through a max-flow computation in a conveniently defined network, we use the following heuristic to find violated generalized subtour breaking constraints (SB). Starting at s, it consists in implementing a Depth First Search in $\hat{G}^s$, to look for cycles of that graph. Whenever a cycle is found, the algorithm checks if a violated inequality (5.16) is implied. The heuristic is run for every satellite $s \in S$.

5.2 Branch-and-cut-and-price algorithm

Over the years Branch-and-price [Barnhart et al., 1998] and Branch-and-cut [Caprara and Fischetti, 1997] algorithms figure among the most successful approaches for solving routing problems. In particular, over the past few years such techniques have been combined into branch-and-cut-and-price algorithms [Fukasawa et al., 2006; Ropke and Cordeau, 2006; Baldacci et al., 2008, 2011], achieving the best results of the literature for their respective problems. In this section, we indicate how our Branch-and-cut-and-price algorithm for 2E-CVRP is implemented.

We first describe how the LP(F2) lower bounds are evaluated by means of Column Generation. Then, we show how valid inequalities (5.13)-(5.16) are separated as cutting planes, allowing the strengthening of bounds LP(F2). Branching rules and additional implementation details are also provided.

5.2.1 Evaluating the LP(F2) Bounds by Column Generation

Assume that a Linear Programming Master Program (LPMP) is obtained from model (5.1)-(5.12) by replacing constraints (5.9) and (5.10) by $\{\lambda_{r_1} \geq 0, r_1 \in R_1\}$ and $\{0 \leq \gamma_{r_2}^s \leq 1, s \in S, r_2 \in \mathcal{R}_2^s\}$, respectively. Assume as well that sets $\mathcal{R}_2^s$ in such a LPMP are replaced by initial subsets of routes $\hat{\mathcal{R}}_2^s \subset \mathcal{R}_2^s : s \in S$ ($|\hat{\mathcal{R}}_2^s| << \mathcal{R}_2^s$), giving rise to a Restricted Linear Programming Master Problem (RLPMP). Recall that, in our

CG procedure, we enumerate all variables and constraints written in terms of routes $r_1 \in R_1$ and keep them in the RLPMP. As such, only $\gamma^s_{r_2}$ variables are priced out, as described next.

Assuming that an optimal basic feasible solution to the RLPMP does exist, let $\hat{\beta} \in \mathbb{R}_-, \{\hat{\tau}_s \in \mathbb{R}_- : s \in S\}, \{\hat{\theta}_i \in \mathbb{R} : i \in C\}$ and $\{\hat{\mu}_s \in \mathbb{R} : s \in S\}$ be optimal basic dual variables, respectively assigned to constraints (5.3), (5.4), (5.5) and (5.6). Whenever for a given $s \in S$, there is a route $r_2 \in \mathcal{R}^s_2 \setminus \hat{\mathcal{R}}^s_2$ such that the dual constraint

$$\hat{\beta} + \hat{\tau}_s + \sum_{i \in C} a_{ir} \hat{\theta}_i + \sum_{i \in C} a_{ir} d_i \hat{\mu}_s \leq c_{r_2} \tag{5.17}$$

is violated, such route is added to $\hat{\mathcal{R}}^s_2$, and a new, enlarged RLPMP is formulated and re-optimized. When constraints (5.17) are satisfied for all $s \in S, r_2 \in \mathcal{R}^s_2 \setminus \hat{\mathcal{R}}^s_2$, the solution to the RLPMP at hands also solves the LPMP. Accordingly, the objective of the LPMP provides a lower bound on the cost of an optimal 2E-CVRP solution. This bound is precisely LP(F2).

The associated pricing problem, that of pricing q-routes, is formulated as a Shortest Path Problem with Capacity Constraints (SPPCC) in a conveniently defined graph. For its resolution, we used a pseudo-polynomial Dynamic Programming algorithm proposed by Christofides et al. [1981]. Roughly speaking, such an algorithm builds paths that start at the depot (in our case, a satellite $s \in S$), visits some vertices in C and returns to s. These paths also respect capacities Q_2. The least expensive of such paths is the solution to the pricing problem. In order to prevent cycles involving only two vertices, we store the cheapest and the second cheapest paths in the course of the algorithm (see Fukasawa et al. [2006] for details). Such a modification does not imply substantial additional computational effort, while allows the bounds to be significantly strengthened.

5.2.2 Strengthening the LP(F2) bounds with cutting planes

We attempt to go beyond bound LP(F2), by strengthening formulation F2 with valid inequalities (5.13)-(5.16), that are violated by the solution to the LPMP implied by F2. That is accomplished by first applying a round of cut generation after solving the LPMP. We separate valid inequalities (5.13)-(5.16) in a specified order (to be described shortly) and after some cuts are appended into the model, re-optimization is conducted. We then attempt to separate additional valid inequalities. Once the separation algorithms fail in providing violated cuts, that specific round of cut generation is finished. Then, another round of column generation is applied. This process of iterat-

ing between row and column generation stops when, neither violated cuts nor columns with negative reduced cost are found by the separation and pricing procedures.

The Branch-and-cut-and-price algorithm uses the procedures in Lysgaard [2004], to separate RC, SC and HM inequalities. The separation algorithm for SB was implemented by ourselves. One important aspect about our implementation is the order in which the separation procedures are called. We first separate RC. Only if no violated RCs are found, we separate the other three valid inequalities. In doing so, we obtained the same lower bound improvement with the inclusion of fewer and sparser cuts.

Assume now that after the re-optimization process that followed the last successful identification of violated cuts, I^{RC}, I^{SC}, I^{HM} and I^{SB} respectively denote the sets of valid inequalities (5.13), (5.14), (5.15) and (5.16) that are appended to the strengthened RLPMP at hands. As before, assume that $\hat{\mathcal{R}}_2^s : s \in S$ are the sets of level-2 variables that are included in the restricted master program. Accordingly, let $\{\hat{\chi}_z \in \mathbb{R}_+ : z \in I^{RC}\}, \{\hat{\rho}_z \in \mathbb{R}_+ : z \in I^{SC}\}, \{\hat{\sigma}_z \in \mathbb{R}_- : z \in I^{HM}\}$ and $\{\hat{\nu}_z \in \mathbb{R}_- : z \in I_s^{SB}, s \in S\}$ be optimal dual basic variables, respectively associated to such inequalities. Define ψ_{ij}^z as an integer parameter that assumes value 0 whether arc $(i,j) \in A_2$ does not belong to inequality $z \in I^{RC} \cup I^{SC} \cup I^{HM} \cup I^{SB}$, otherwise it assumes the coefficient of such arc on the inequality. The pricing algorithm consists in finding a route $r_2 \in \mathcal{R}_2^s \setminus \hat{\mathcal{R}}_2^s$ that violates the constraint

$$\beta+\tau_s+\sum_{i\in C} a_{ir}\theta_i+\sum_{i\in C} a_{ir}d_i\mu_s+\sum_{(i,j)\in A_2\,:\,b_{ijr_2}=1}\Big(\sum_{z\in I^{RC}}\psi_{ij}^z\chi_z+\sum_{z\in I^{SC}}\psi_{ij}^z\rho_z+\sum_{z\in I^{HM}}\psi_{ij}^z\sigma_z+\sum_{z\in I_s^{SB}}\psi_{ij}^z\nu_z\Big)\le c_{r_2}. \tag{5.18}$$

It is important to point out that the inclusion of inequalities (5.13)-(5.16) in the model do not change the structure of the pricing problems. The dual variables assigned to (5.13)-(5.16) only aftect arc costs in the associated SPPCC and, in this sense, the Branch-and-cut-and-price algorithm is robust [Fukasawa et al., 2006]. As such, the Dynamic Programming algorithm discussed previously can be used for solving the pricing problems, even after the addition of cuts to the model.

In our implementation, a good balance between lower bound improvement due to the addition of cuts and their separation cost was obtained by the following strategy. Cut generation is only conducted at the root node and in other nodes whose height in the search tree is a multiple of an integer implementation parameter d (in our case, we set $d = 6$). At the root node, all violated inequalities found by the separation algorithms are appended in the model. In the other nodes, only the most violated in each family is included. Other policies that we have experimented with are: the addition of cuts at every node and the addition of cuts only at the root node. Both, however, were outperformed in terms of CPU times, by the mixed strategy.

5.2.3 Branching rules

Let $\{\hat{\lambda}_{r_1} : r_1 \in R_1\}$ and $\{\hat{\gamma}^s_{r_2} : s \in S, r_2 \in \hat{\mathcal{R}}^s_2\}$ denote the solution to the last RLPMP formulated and solved after the last round of column and row generation was applied. If it is integer feasible, it also solves the IP (5.1)-(5.12). Being fractional, we resort to branching. Note that a solution to RLPMP that involves q-routes that do not satisfy elementarity cannot be integer.

Preference is given to branch first, on fractional λ_{r_1} variables, since all these variables are always explicitly considered in the RLPMP. The algorithm branches on the most fractional variable. Assuming that $\hat{\lambda}_w$ for a given $w \in R_1$ is the most fractional value, two child nodes are created: one assigned to $\lambda_w \leq \lfloor \hat{\lambda}_w \rfloor$ and the other assigned to $\lambda_w \geq \lceil \hat{\lambda}_w \rceil$.

If $\{\hat{\lambda}_{r_1} : r_1 \in R_1\}$ are all integer valued, a second type of branching constraints are imposed. First, we evaluate the total number $\sum_{r_2 \in \hat{\mathcal{R}}^s_2} \hat{\gamma}^s_{r_2}$ of vehicles that serves each satellite $s \in S$ at the solution to the RLPMP. Assuming that $\hat{s}$ is the satellite that uses the most fractional number of vehicles, two child nodes are created: one assigned to $\sum_{r_2 \in \mathcal{R}^{\hat{s}}_2} \gamma^{\hat{s}}_{r_2} \leq \lfloor \sum_{r_2 \in \hat{\mathcal{R}}^{\hat{s}}_2} \hat{\gamma}^{\hat{s}}_{r_2} \rfloor$ and the other to $\sum_{r_2 \in \mathcal{R}^{\hat{s}}_2} \gamma^{\hat{s}}_{r_2} \geq \lceil \sum_{r_2 \in \hat{\mathcal{R}}^{\hat{s}}_2} \hat{\gamma}^{\hat{s}}_{r_2} \rceil$.

The third branching strategy is imposed if $\{\hat{\lambda}_{r_1} : r_1 \in R_1\}$ and $\{\sum_{r_2 \in \hat{\mathcal{R}}^s_2} \hat{\gamma}^s_{r_2} : s \in S\}$ are all integer. It consists in branching on level-2 arcs. Assuming that $(p, q) \in A_2$ denotes the arc for which the amount $\sum_{r_2 \in \hat{\mathcal{R}}^s_2} b_{pqr_2} \hat{\gamma}^s_{r_2}$ is the most fractional among all arcs, two nodes are created. In one of them, we impose $\sum_{r_2 \in \mathcal{R}^s_2} b_{pqr_2} \gamma^s_{r_2} = 1$. In the other, $\sum_{r_2 \in \mathcal{R}^s_2} b_{pqr_2} \gamma^s_{r_2} = 0$ is enforced.

5.2.4 Further implementation details

Let us now address how the first RLPMP is formulated, i.e., how the initial sets of routes ($\bigcup_{s \in S} \hat{\mathcal{R}}^s_2$) are chosen. For each $s \in S$, we randomly add unattended customers to the route that starts and ends in s, until no additional customers could be added, without violating capacity Q_2. The procedure is repeated for every possible satellite, until all customers are visited by precisely one level-2 route.

Our pricing algorithm evaluates routes on sets $\hat{\mathcal{R}}^s_2$ quite fast. However, such sets comprise a large number of routes. As a consequence, the RLPMP becomes dense along the branch-and-bound tree, because a large number of variables is appended during the CG iterations. To deal with such a problem, we introduce a policy to drop variables from RLPMP whenever such variables are no longer useful. For each CG iteration we extract from RLPMP the dual cost associated with variables $\gamma^s_{r_2}$. Those variables on which the dual cost is greater than d for more than n iterations are dropped from the model. Note that, such variables may be priced again on next RLPMP. However, this

behavior is obviously no longer desired, cause it would increase the computational cost. In our implementation, we set parameters $d = 50$ and $n = 20$.

Aiming at providing (hopefully good) 2E-CVRP upper bounds, we embed a Column Generation Heuristic (CGH) in the algorithm. The procedure consists in solving, at certain nodes of the enumeration tree, an IP like (5.1)-(5.12), formulated in terms of the sets of columns $\bigcup_{s \in S} \hat{\mathcal{R}}_2^s$ that were priced out when the last RLPMP was solved at that node. Routes that does not satisfy the elementarity condition are not included in the restricted IP. We solve each restricted IP to proven optimality by a Branch-and-bound algorithm. A time limit of 100 seconds was imposed for the entire CPU time dedicated to CGH.

The algorithm investigates nodes on the branch-and-bound tree using the best first search strategy. In order to solve each RLPMP, `CPLEX` LP commercial solver (release 12.1) was used. `CPLEX` branch-and-bound is used to solve each restricted IP formulated during the application of `CGH`.

5.3 Computational Experiments

In this section, we present computational results obtained with BCP and compare its results to those provided by three other algorithms in the literature: the two Branch-and-price implementations in our previous study [Santos et al., 2012a], BP-NE and BP-E, and the Branch-and-cut algorithms in Perboli et al. [2011] and Jepsen et al. [2012], named respectively BC-PER and BC-JEP.

5.3.1 Test instances

Computational testings were conducted with four sets of test instances (set 1,..., set 4). Instances in sets 1 to 3 were generated from instances E-n13-k4, E-n22-k4, E-n33-k4 and E-n51-k5 introduced for CVRP by Christofides and Eilon [1969]. For each of these CVRP instances, several 2E-CVRP instances were generated as follows.

Set 1 comprises instances generated from E-n13-k4 and includes 13 vertices (in the CVRP case, 12 customers besides the depot). Each instance in set 1 involves two satellites, twelve customers and the depot. To generate 2E-CVRP set 1 instances from E-n13-k4, all possible $66 = \binom{12}{2}$ combinations of two satellites out of the 12 vertices were considered. Set 2 and set 3 were generated in a similar fashion, from E-n22-k4, E-n33-k4 and E-n51-k5. That was accomplished by choosing two and four satellites, for each instance in sets 2 and 3, respectively. For example, instance E-n22-k4-s6-17 is the 2E-CVRP set-2 instance obtained when vertices 6 and 17 of E-n22-k4 were chosen

to play the role of satellites. Similarly, E-n51-k5-2-4-17-46 was obtained from E-n51-k5 when $S = \{2, 4, 17, 46\}$.

The fourth test set was proposed by Crainic et al. [2010] and comprises instances with 50 customers and 2, 3 or 5 satellites. In total, our test bed involves 161 instances. They are all available for download from Beasley [1990] and, to the best of our knowledge, span the entire set of instances in the 2E-CVRP literature. For all instances, $L_s = 0, s \in S$.

5.3.2 Computational results

BCP was coded in `C++` and computational experiments were conducted with an Intel Core i7 machine running at 3.3 GHz with 4 GBytes of RAM memory, under Linux Operating System. `gcc` compiler was used, with optimization flag `-o` turned on. Computational results reported here for BP-NE and BP-E were extracted from Santos et al. [2012a]. Experiments in our earlier study were conducted in the same computational environment used here to test BCP. Computational results reported here for BC-PER were extracted from Perboli et al. [2011]. They were obtained with a 3Ghz Intel Pentium 4 machine. For BC-JEP results were obtained from Jepsen et al. [2012] using a Intel Xeon X5550 2.67 GHz with 24 GB of memory. Although the computational environments used to test BCP, BC-PER and BC-JEP are different, the order of magnitude of the CPU times taken by both methods can be compared.

Let us first discuss how the LP lower bounds available for 2E-CVRP in the literature do compare to each other. To that aim, we use the following notation to refer to other formulations considered here:

- F2: formulation (5.1)-(5.12) found at the root node of our Branch-and-cut-and-price algorithm.

- F2$^+$: bounds from F2 strengthened with valid inequalities (5.13)-(5.16). We recall that, inequalities (5.13)-(5.16) are separated heuristically. Therefore, the lower bound from F2$^+$ may be not the true bound that would be implied if all inequalities (5.13)-(5.16) were added to F2.

- F1^{NE}: the formulation used by BP-NE in Santos et al. [2012a]. That formulation also considers q-routes, but in a formulation that is weaker than F2.

- F1^E: the formulation used by BP-E in Santos et al. [2012a]. F1^E considers level-2 routes that satisfy elementarity.

- PER: the formulation behind BC-PER, the Branch-and-cut algorithm in Perboli et al. [2011]. It is a network flow formulation strengthened with two classes of valid inequalities.

- JEP: the formulation behind BC-JEP, the Branch-and-cut algorithm in Jepsen et al. [2012]. It provides a relaxed formulation for 2E-CVRP by integrating SDCVRP and the Capacitated Location Routing Problem. Such a formulation provides valid lower bounds and feasible solutions that may be accepted after a checking procedure.

In Table 5.1, we report on 2E-CVRP lower bounds. For each instance in sets 2 and 4, we provide the LP bound implied by each formulation and the respective CPU time, labeled as t(s), spent do evaluate it. We first report the instance name, followed by LP and t(s) for F2, F2$^+$, F1^{NE}, F1^E. Then, we report only the LP for PER and JEP. t(s) values for such approaches are not reported cause they were not reported in their original paper. Similarly, lower bounds of set-3 instances were not reported cause they are not available in Perboli et al. [2011].

Results in Table 5.1 confirm that LP bounds provided by model F2 are stronger than those provided by model F1 [Santos et al., 2012a]. The replacement of the binary variables (indexed by vehicles) used in model F1 by new integer constrained variables allowed F2 to include tighter coupling constraints between level 1 and level 2 variables and tighter level 1 capacity constraints (constraints (5.6) and (5.7), respectively). That modification alone allowed LP(F2) to be, in some cases, almost 14% stronger than LP(F1).

In addition, valid inequalities (5.13)-(5.16) help F2$^+$ to provide the strongest bounds for the most instances of the literature. For instances indicated in Table 5.1, LP(F2$^+$) are, on the average, 7.3% stronger than LP(F2) counterparts. On the average, bounds LP(F2$^+$) are 5.2% stronger than LP(F1^E). LP(F2$^+$) are always stronger than LP(PER) and LP(JEP), for all set-2 and set-4 instances (except for instance n22-s8-14). On the average, for instances indicated in Table 5.1, LP(F2$^+$) are 7.76% stronger than LP(PER) and 6.27% stronger than LP(JEP).

Considering the 21 set-2 instances indicated in Table 5.1, LP(F1^E) bounds are stronger than LP(F2$^+$) in 8 cases. The opposite is true for 10 instances. However, for instances in set-4, the hardest in our test bed, LP(F2$^+$) is always stronger than LP(F1^E). Noteworthy is the fact that the CPU time needed to evaluate LP(F2$^+$) are always smaller. Quite often, the times needed to evaluate them are two or three orders of magnitude smaller than the counterparts for LP(F1^E).

	F2		$F2^+$		$F1^{NE}$		$F1^E$		PER	JEP
instance	LP	t(s)	LP	t(s)	LP	t(s)	LP	t(s)	LP	LP
set 2										
n22-s6-17	411.9	0.05	412.3	0.09	411.9	0.06	412.3	2.4	411.1	403.1
n22-s8-14	367.5	0.04	375.7	0.11	367.5	0.05	374.9	2.1	369.9	377.2
n22-s9-19	450.2	0.04	465.3	0.13	450.2	0.06	456.9	1.9	441.1	425.5
n22-s10-14	360.5	0.05	366.1	0.11	360.5	0.06	366.0	2.7	360.5	359.5
n22-s11-12	400.8	0.04	410.4	0.10	400.8	0.06	410.4	1.7	395.7	404.5
n22-s12-16	378.3	0.04	391.9	0.15	378.3	0.05	392.1	1.8	366.3	376.7
n33-s1-9	703.4	1.39	719.8	2.97	703.4	1.92	717.1	24	696.7	637.7
n33-s2-13	685.2	1.29	701.6	2.94	685.2	1.76	699.9	46	675.8	638.6
n33-s3-17	655.9	1.82	674.9	3.74	655.9	2.07	673.9	32	657.3	644.1
n33-s4-5	706.2	2.53	725.0	3.67	706.2	2.94	740.7	284	713.8	682.3
n33-s7-25	718.0	1.66	730.7	2.79	718.0	1.98	736.5	26	718.3	650.5
n33-s14-22	751.8	1.57	770.2	4.42	751.8	1.86	766.0	75	750.9	685.5
n51-s2-17	553.6	1.01	559.7	2.61	553.6	1.44	562.0	960	542.6	553.3
n51-s4-46	511.3	1.09	520.1	3.11	511.3	1.29	518.1	1216	509.3	514.6
n51-s6-12	522.3	1.34	528.4	2.66	522.3	1.57	530.8	1306	510.4	523.7
n51-s11-19	548.7	1.11	555.5	2.52	548.7	1.34	556.1	951	551.0	548.1
n51-s27-47	513.5	1.08	518.6	3.29	513.5	1.19	518.3	468	505.8	513.8
n51-s32-37	525.4	1.03	531.0	2.52	525.4	1.30	529.0	525	517.3	528.8
n51-s2-4-17-46	493.1	1.22	505.9	5.78	487.0	1.74	506.4	1440	503.6	500.9
n51-s6-12-32-37	495.2	1.13	503.9	3.01	495.2	1.66	508.6	1080	501.8	503.0
n51-s11-19-27-47	500.0	1.13	506.1	4.34	499.8	1.59	506.3	1432	500.4	504.1
set 4										
Instance50-s5-37	1378.9	39	1411.3	73	1289.7	80	1335.0	2086	1259.5	1262.6
Instance50-s5-38	1041.5	68	1070.2	173	1004.8	93	1054.7	562	972.8	965.6
Instance50-s5-39	1381.3	42	1399.9	70	1265.8	65	1302.6	981	1239.7	1286.1
Instance50-s5-40	1022.3	62	1045.7	149	950.4	100	998.6	514	970.8	977.4
Instance50-s5-41	1546.8	43	1583.5	78	1470.1	69	1510.0	1533	1356.8	1445.2
Instance50-s5-42	1084.3	60	1125.9	150	1055.5	90	1118.4	998	1000.0	995.2
Instance50-s5-43	1234.9	37	1333.4	100	1158.4	59	1272.0	1544	1124.1	1258.7
Instance50-s5-44	892.5	61	943.5	151	860.3	119	921.0	2245	834.9	840.0
Instance50-s5-45	1232.5	35	1304.5	86	1153.8	72	1245.1	1365	1118.2	1190.5
Instance50-s5-46	866.5	63	915.0	125	815.8	96	889.9	1039	853.9	843.0
Instance50-s5-47	1426.4	43	1497.4	108	1360.1	76	1456.2	3594	1230.7	1324.9
Instance50-s5-48	974.7	62	1042.6	149	949.2	95	1028.5	1394	893.3	931.7
Instance50-s5-49	1311.6	43	1345.1	92	1212.8	69	1245.0	547	1196.2	1209.4
Instance50-s5-50	943.0	56	971.0	133	891.3	98	924.0	421	879.1	843.1
Instance50-s5-51	1243.3	42	1270.6	91	1145.1	75	1175.1	653	1117.0	1150.0
Instance50-s5-52	942.9	60	971.4	143	873.8	110	910.7	335	895.0	911.7
Instance50-s5-53	1441.3	43	1489.0	107	1368.2	76	1408.5	1162	1240.2	1313.8
Instance50-s5-54	1024.3	66	1057.3	166	988.2	122	1019.6	734	928.9	994.6

Table 5.1: Comparison of Linear Programming lower bounds and respective CPU times for 2E-CVRP - instances in sets 2 and 4.

Let us now address the capabilities of BCP in solving small instances, namely, those in the first set. All such instances could be solved to proven optimality by previous methods. In Table 5.2, we compare BCP, BP-NE and BC-PER. For each instance we first report the optimal objective function (opt), followed by the CPU times taken by BCP, BP-NE (the fastest algorithm in Santos et al. [2012a] for set 1) and BC-PER. Results for BC-JEP are not in Table 5.2 cause they were not reported by the authors. In addition, results for BP-E were omitted from the table, since they are dominated by BP-NE for all instances of set 1.

BCP clearly dominates previous approaches for instances in set 1. While the total CPU time BCP needed to solve all instances in this set is 17.8 seconds, BC-PER and BP-NE took 236.6 and 3674.9 seconds, respectively.

In Tables 5.3 and 5.4, we compare BCP, BP, BC-PER and BC-JEP for solving larger instances (sets 2 and 3 with up to 51 customers and 4 satellites). A maximum CPU time limit of 10000 seconds was imposed for the execution of BCP, BP, BC-PER and BC-JEP. For each algorithm, we report the best lower (BLB) and upper (BUB) bounds obtained at the end of the search. For BCP, we also provide the number of Branch-and-bound nodes investigated and the number of cuts added during the search.

Since for those instances left unsolved BP-NE provides better lower bounds than BP-E, while the opposite is true for the upper bounds, in Tables 5.3 and 5.4, we report in columns under headings BP, the results for the combination of the best results obtained by BP-E and BP-NE. The same applies for BC-PER. In Perboli and Tadei [2010], the Branch-and-cut algorithm separates valid inequalities for CVRP, while in Perboli et al. [2008a, 2011] they are not separated. Therefore, in columns under headings BC-PER in Tables 5.3-5.4, we report the best lower and upper bounds, obtained by the combination of results in Perboli et al. [2008a]; Perboli and Tadei [2010]; Perboli et al. [2011].

BCP and BC-JEP managed to provide optimality certificates for 18 out of 21 instances of set 2, while BP and BC-PER respectively solved 8 and 13 instances. Although BCP and BC-JEP was capable of solving the same number of instances, the average duality gap attained at termination for the other 3 instances of BCP is smaller than BC-JEP: 0.30% against 2.38%. The maximum duality gap attained at termination was 0.7%, 5.7%, 4.6% and 4.7% for BCP, BP-NE, BC-PER and BC-JEP, respectively.

Set 2 and set 3 were generated from the same CVRP instances, but differ in terms of the sets of chosen satellites. Because Perboli et al. [2011] and Jepsen et al. [2012] select satellites for instances with 51 customers of different ways, we report on table 5.4 only those results for instances with 22 and 33 customers. For such instances BCP,

satellites	opt	CPU time BCP	BP-NE	BC	satellites	opt	CPU time BCP	BP-NE	BC
1,2	280	0.9	58.1	835	**4,8**	252	0.1	0.5	0.6
1,3	286	0.7	4.4	247	**4,9**	264	0.1	0.4	3.7
1,4	284	1.1	64.0	251	**4,10**	272	0.5	1.9	3.5
1,5	218	0.4	0.7	1.6	**4,11**	296	0.2	1.2	5.0
1,6	218	0.3	0.6	2.0	**4,12**	304	0.2	0.7	5.7
1,7	230	0.4	1.1	5.3	**5,6**	248	0.1	0.7	3.6
1,8	224	0.2	0.8	2.7	**5,7**	254	0.2	0.5	2.6
1,9	236	0.2	0.4	6.2	**5,8**	256	0.1	0.6	4.0
1,10	244	0.2	0.9	5.1	**5,9**	262	0.3	0.7	4.9
1,11	268	0.3	0.7	7.8	**5,10**	262	0.2	0.6	2.8
1,12	276	0.1	1.5	28.7	**5,11**	262	0.2	0.6	1.4
2,3	290	0.3	5.6	392	**5,12**	262	0.1	0.5	1.7
2,4	288	0.4	22.3	639	**6,7**	280	0.4	0.9	19.3
2,5	228	0.2	0.6	2.5	**6,8**	274	0.2	0.6	5.4
2,6	228	0.3	0.7	12.2	**6,9**	280	0.1	1.0	11.0
2,7	238	0.3	0.9	3.4	**6,10**	280	0.1	0.8	9.1
2,8	234	0.1	0.8	4.0	**6,11**	280	0.2	0.7	4.8
2,9	246	0.1	0.6	6.9	**6,12**	280	0.2	1.2	4.3
2,10	254	0.5	3.8	7.4	**7,8**	292	0.2	0.7	1.8
2,11	276	0.6	0.9	9.6	**7,9**	300	0.3	0.5	7.9
2,12	286	0.5	1.4	136	**7,10**	304	0.3	0.9	14.1
3,4	312	1.3	103	810	**7,11**	310	0.4	1.1	6.8
3,5	242	0.2	0.8	2.2	**7,12**	310	0.7	0.9	8.6
3,6	242	0.2	0.6	1.8	**8,9**	326	0.1	0.8	16.4
3,7	252	0.3	0.9	1.6	**8,10**	326	0.3	0.6	15.1
3,8	248	0.1	1.0	1.9	**8,11**	326	0.2	1.0	5.2
3,9	260	0.3	0.8	3.4	**8,12**	326	0.2	0.9	4.9
3,10	268	0.5	1.7	8.3	**9,10**	338	0.3	1.6	17.1
3,11	290	0.4	2.0	5.7	**9,11**	350	0.1	0.9	17.8
3,12	300	0.3	1.6	7.2	**9,12**	350	0.3	0.7	10.8
4,5	246	0.1	0.5	2.8	**10,11**	358	0.5	0.6	55.1
4,6	246	0.1	0.8	2.9	**10,12**	358	0.4	1.0	21.3
4,7	258	0.1	0.7	5.3	**11,12**	400	0.2	1.4	34.0

Table 5.2: CPU times (in seconds) taken by BCP, BP-NE and BC to solve each set 1 instance to proven optimality.

BP, BC-PER and BC-JEP could be compared. BCP managed to solve all 12 instances with 22 and 33 customers and BC-JEP was capable of solving 11 instances. BP and BC-PER perform poorly and could solve only 6 instances with 22 customers.

Instances of set 4 were proposed by Crainic et al. [2010] to span different distribution scenarios on 2E-CVRP. They consider 50 customers with $2, 3$ or 5 satellites distributed along the euclidean plane using different strategies. Perboli et al. [2011] and Jepsen et al. [2012] present their results differently for instances of set 4. On the

	BCP				BP		BC-PER		BC-JEP	
instance	nodes	cuts	BLB	BUB	BLB	BUB	BLB	BUB	BLB	BUB
n22-s6-17	3	16	417.0	417.0	417.0	417.0	417.0	417.0	417.0	417.0
n22-s8-14	5	27	384.9	384.9	384.9	384.9	384.9	384.9	384.9	384.9
n22-s9-19	197	46	470.6	470.6	470.6	470.6	470.6	470.6	470.6	470.6
n22-s10-14	3	34	371.5	371.5	371.5	371.5	371.5	371.5	371.5	371.5
n22-s11-12	13	33	427.2	427.2	427.2	427.2	427.2	427.2	427.2	427.2
n22-s12-16	3	44	392.7	392.7	392.7	392.7	392.7	392.7	392.7	392.7
n33-s1-9	5184	230	730.1	730.1	722.3	730.1	730.1	730.1	730.1	730.1
n33-s2-13	2217	178	714.6	714.6	703.3	714.6	714.6	714.6	714.6	714.6
n33-s3-17	4989	278	707.4	707.4	680.2	707.4	707.4	707.4	707.4	707.4
n33-s4-5	6497	221	778.7	778.7	756.3	785.2	778.7	778.7	778.7	778.7
n33-s7-25	6049	312	755.6	756.8	747.3	756.8	756.8	756.8	756.8	756.8
n33-s14-22	23	107	779.0	779.0	775.0	779.0	779.0	779.0	779.0	779.0
n51-s2-17	9278	379	597.0	597.4	562.9	597.4	582.2	597.5	570.4	597.4
n51-s4-46	134	115	530.7	530.7	530.7	530.7	529.3	530.7	530.7	530.7
n51-s6-12	8387	374	550.9	554.8	537.2	557.2	541.1	554.8	545.7	554.8
n51-s11-19	6502	302	581.6	581.6	576.7	591.5	559.8	581.6	581.6	581.6
n51-s27-47	6043	361	538.2	538.2	524.5	538.2	535.0	538.2	534.1	538.2
n51-s32-37	8229	369	552.2	552.2	539.5	552.2	552.2	552.2	552.2	552.2
n51-s2-4-17-46	421	227	530.7	530.7	530.7	530.7	515.7	541.0	530.7	530.7
n51-s6-12-32-37	2148	342	531.9	531.9	520.6	531.9	516.0	538.8	531.9	531.9
n51-s11-19-27-47	3539	359	527.6	527.6	523.0	527.6	519.5	531.2	527.6	527.6

Table 5.3: Comparison of BCP, BP, BC-PER and BC-JEP - set 2.

	BCP				BP		BC-PER		BC-JEP	
instance	nodes	cuts	BLB	BUB	BLB	BUB	BLB	BUB	BLB	BUB
E-n22-k4-s13-14	111	31	526.1	526.1	526.1	526.1	526.1	526.1	526.1	526.1
E-n22-k4-s13-16	147	38	521.0	521.0	521.0	521.0	521.0	521.0	521.0	521.0
E-n22-k4-s13-17	5	32	496.3	496.3	496.3	496.3	496.3	496.3	496.3	496.3
E-n22-k4-s14-19	17927	287	498.8	498.8	498.8	498.8	498.8	498.8	498.8	498.8
E-n22-k4-s17-19	295	64	512.8	512.8	512.8	512.8	512.8	512.8	512.8	512.8
E-n22-k4-s19-21	307	57	520.4	520.4	520.4	520.4	520.4	520.4	520.4	520.4
E-n33-k4-s16-22	9093	238	672.1	672.1	641.7	674.7	634.2	672.1	657.9	676.1
E-n33-k4-s16-24	2189	146	666.0	666.0	650.2	669.1	625.7	668.8	666.0	666.0
E-n33-k4-s19-26	5136	259	680.3	680.3	662.0	680.3	648.2	680.3	680.3	680.3
E-n33-k4-s22-26	6836	288	680.3	680.3	661.5	680.3	652.1	680.7	680.3	680.3
E-n33-k4-s24-28	709	94	670.4	670.4	663.0	670.4	633.6	672.3	670.4	670.4
E-n33-k4-s25-28	6312	200	650.5	650.5	637.8	651.3	616.5	651.2	650.5	650.5

Table 5.4: Comparison of BCP, BP, BC-PER and BC-JEP - set 3.

one hand, Perboli et al. [2011] report only those results for instances with 5 satellites. They also set parameter $m_s = K_2$, which means that the maximum number of vehicles allowed to start from a given satellite s is unbounded. On the other hand, Jepsen et al.

[2012] report their results for all instances of set 4 (54 instances). In addition, they impose a limit on the number of vehicles leaving satellites, setting parameter $m_s < K_2$. Because of that, in the following we split our comparisons. On Table 5.5 we report results comparing BCP, BP and BC-PER, while on Table 5.7 we compare BCP, BP and BC-JEP. Entries on such tables are the same reported for sets 2 and 3 above.

	BCP				BP		BC-PER	
instance	nodes	cuts	BLB	BUB	BLB	BUB	BLB	BUB
Instance50-s5-37	1058	218	1512.0	1545.5	1463.8	1542.3	1434.5	1548.0
Instance50-s5-38	429	235	1143.5	1194.3	1093.4	1228.3	1076.4	1185.5
Instance50-s5-39	1054	280	1508.9	1528.8	1477.6	1533.2	1423.4	1525.2
Instance50-s5-40	432	406	1144.7	1207.7	1066.8	1221.5	1068.5	1199.4
Instance50-s5-41	790	267	1643.5	1659.6	1609.2	1698.4	1580.8	1703.0
Instance50-s5-42	553	318	1172.4	1230.4	1130.4	1265.2	1097.8	1223.0
Instance50-s5-43	843	329	1394.8	1427.0	1321.5	1456.9	1341.0	1453.1
Instance50-s5-44	464	246	1017.5	1089.2	937.8	1162.4	935.4	1039.3
Instance50-s5-45	918	354	1386.9	1429.5	1348.9	1495.8	1331.5	1469.0
Instance50-s5-46	526	262	1044.3	1124.6	916.1	1103.0	930.5	1095.6
Instance50-s5-47	640	460	1543.5	1586.0	1491.7	1608.2	1487.0	1598.8
Instance50-s5-48	485	285	1050.1	1097.8	1005.3	1108.6	998.6	1096.9
Instance50-s5-49	704	263	1432.8	1434.8	1375.5	1496.1	1370.8	1479.1
Instance50-s5-50	582	231	1029.4	1104.7	981.2	1119.0	968.4	1090.6
Instance50-s5-51	882	254	1370.7	1398.0	1360.9	1429.1	1310.9	1436.3
Instance50-s5-52	562	272	1094.2	1131.6	1012.2	1140.5	1003.0	1128.3
Instance50-s5-53	576	333	1529.4	1551.5	1501.7	1614.2	1483.1	1552.7
Instance50-s5-54	496	217	1101.6	1143.0	1071.2	1156.0	1034.8	1135.3

Table 5.5: Comparison of BCP, BP and BC-PER - set 4.

Set-4 instances are the hardest to solve by a combination of two mains aspects. Firstly, they involve up to 5 satellites. In addition, vehicles have larger capacities, which slows down the Dynamic Programming algorithm that prices q-routes. That explains why, compared to sets 2 and 3 instances for example, BCP evaluated much fewer Branch-and-bound nodes within the same time limit. Because Perboli et al. [2011] consider only those instances of set 4 with 5 satellites and neglect the upper bound on the number of vehicles starting from a given satellite, all such instances remain unsolved. For them, the average duality gaps attained at termination are quite similar for BC-PER and BP. While BC-PER provides better upper bounds, BP evaluates stronger lower bounds. BCP outperforms both algorithms. On the one hand, it provided the best lower bounds. On the other hand, it provided the best upper bounds for 9 out of 18 instances.

For those 54 instances of set 4 considered by Jepsen et al. [2012], 15 instances could be solved to proven optimality by BC-JEP, while only 3 could be solved by BCP.

However, for those instances left unsolved at termination of time limit, BCP presents smaller duality gaps than BP and BC-JEP. The average duality gap for such instances is respectively of $1.35\%, 9.14\%$ and 7.2% for BCP, BP and BC-JEP. In addition, the maximum duality gap achieved by BCP at termination is 5.12%, while for BP is 24.6% and for BC-JEP is 14.73%.

A comparison BCP, BC-PER and BP-JEP is summarized in Table 5.6. BP is omitted from table cause it is dominated by BCP for all instances of our test set. In the first line of that table, we provide the number of instances in each test set. We then report, in the next line, the number of instances in each set that were solved to proven optimality by each algorithm. In the third line, we report the average CPU time (in seconds) needed by each algorithm, considering only those instances each algorithm solved to optimality. In the last line, we report the average duality gap $(100(BUB - BLB)/(BUB))$, considering the instances each algorithm could not solve within the time limit.

	set2			set3			set4 ($m_s = K_2$)		set4 ($m_s < K_2$)	
	BCP	BC-PER	BC-JEP	BCP	BC-PER	BC-JEP	BCP	BC-PER	BCP	BC-JEP
# total		21			12			18		54
# solved	18	13	18	12	6	11	0	0	3	15
avg time(s)	1413.6	N/A	457.8	1887.6	N/A	94.2	-	-	6748	2173
avg gap(%)	0.3	2.55	2.38	-	5.02	2.78	3.53	9.6	1.35	7.2

Table 5.6: Aggregate results for BCP, BC-PER and BC-JEP.

5.4 Concluding Remarks

Multi-echelon systems have received attention from the researchers and practitioners of the optimization community in the last few years due to its importance on real distribution problems. A special case of multi-echelon system is the two-echelon, which consists in shipping demands from the source to the destination using an intermediate satellite.

In this chapter we dealt with a recent problem in the literature: the Two-Echelon Vehicle Routing Problem (2E-CVRP) which couples routing decisions with a two-echelon system. The 2E-VRP was proposed by Perboli et al. [2008b] and since then have received attention of researchers [Crainic et al., 2009; Perboli and Tadei, 2010; Perboli et al., 2011; Santos et al., 2012a]. Among solutions available for the 2E-VRP we have branch-and-cut, branch-and-price and heuristic algorithms. In this chapter we propose an Integer Programming reformulation and a Branch-and-cut-and-price al-

gorithm for solving the 2E-CVRP. Our algorithm is able to provide new optimality certificates for 10 instances of the literature and also to evaluate stronger lower and upper bounds for those instances left unsolved.

	BCP				BP		BC-JEP	
instance	nodes	cuts	BLB	BUB	BLB	BUB	BLB	BUB
Instance50-s2-1	1844	252	1560.2	1579.5	1516.6	1637.2	1542.8	1771.7
Instance50-s2-2	619	199	1432.4	1442.1	1366.8	1513.7	1438.3	1438.3
Instance50-s2-3	1614	202	1560.5	1586.7	1514.9	1598.4	1545.5	1769.4
Instance50-s2-4	704	236	1419.3	1429.4	1376.7	1460.0	1411.5	1424.0
Instance50-s2-5	1546	181	2183.0	2204.1	2146.2	2204.1	2185.2	2201.0
Instance50-s2-6	587	251	1279.8	1279.8	1248.2	1301.4	1279.8	1279.8
Instance50-s2-7	1904	262	1444.4	1491.7	1339.6	1498.3	1436.5	1665.4
Instance50-s2-8	993	163	1357.8	1369.5	1232.4	1374.3	1363.7	1363.7
Instance50-s2-9	1395	205	1442.4	1454.8	1351.5	1531.2	1431.4	1652.3
Instance50-s2-10	884	166	1396.4	1415.5	1280.3	1427.9	1407.6	1407.6
Instance50-s2-11	1455	197	2034.1	2078.0	1950.0	2106.8	2035.7	2065.4
Instance50-s2-12	802	209	1209.4	1209.4	1127.2	1290.5	1209.4	1209.4
Instance50-s2-13	1730	218	1475.1	1481.8	1406.3	1481.8	1464.1	1657.4
Instance50-s2-14	1012	144	1385.9	1393.6	1301.9	1455.4	1392.6	1416.2
Instance50-s2-15	1687	294	1480.5	1497.7	1411.9	1562.3	1473.7	1666.1
Instance50-s2-16	981	104	1381.3	1390.9	1317.6	1481.3	1389.1	1389.1
Instance50-s2-17	1578	217	2084.5	2090.4	2027.9	2108.9	2086.6	2096.1
Instance50-s2-18	796	122	1221.4	1229.5	1181.1	1229.5	1227.6	1227.6
Instance50-s3-19	1568	284	1558.4	1566.8	1531.8	1588.6	1557.1	1717.3
Instance50-s3-20	1161	193	1266.4	1304.9	1193.2	1376.2	1265.7	1484.4
Instance50-s3-21	1931	308	1567.1	1577.8	1538.9	1577.8	1570.8	1684.5
Instance50-s3-22	461	260	1280.0	1281.8	1250.5	1304.5	1281.3	1281.3
Instance50-s3-23	1781	195	1804.9	1807.3	1761.5	1857.0	1626.1	1807.3
Instance50-s3-24	743	127	1281.4	1282.6	1251.7	1451.3	1282.6	1282.6
Instance50-s3-25	1525	214	1496.2	1534.6	1413.1	1534.6	1503.8	1534.2
Instance50-s3-26	914	197	1154.2	1216.5	1078.4	1244.2	1167.4	1167.4
Instance50-s3-27	1387	188	1457.8	1503.6	1375.1	1503.6	1467.4	1584.8
Instance50-s3-28	1005	201	1199.5	1210.4	1120.3	1301.4	1210.4	1210.4
Instance50-s3-29	1386	284	1708.9	1737.2	1624.1	1744.2	1708.5	1737.2
Instance50-s3-30	604	108	1211.5	1211.5	1128.7	1290.6	1211.5	1211.5
Instance50-s3-31	1672	159	1473.2	1503.9	1412.5	1503.9	1468.4	1635.3
Instance50-s3-32	597	102	1194.1	1225.5	1133.9	1257.4	1192.8	1235.5
Instance50-s3-33	1277	202	1482.3	1516.8	1431.5	1563.2	1488.9	1587.5
Instance50-s3-34	912	176	1226.8	1238.5	1186.6	1249.3	1233.2	1233.9
Instance50-s3-35	1855	165	1707.1	1719.1	1640.5	1728.7	1696.9	1723.5
Instance50-s3-36	847	205	1221.5	1230.8	1181.7	1304.5	1228.8	1228.8
Instance50-s5-37	1402	319	1513.9	1535.8	1480.3	1572.3	1484.4	1661.1
Instance50-s5-38	403	274	1164.9	1192.6	1094.7	1217.8	1167.3	1265.4
Instance50-s5-39	1704	191	1512.9	1520.9	1464.0	1520.91	1515.1	1520.9
Instance50-s5-40	718	186	1190.2	1199.4	1080.1	1219.3	1168.6	1211.3
Instance50-s5-41	1352	320	1657.0	1676.2	1595.3	1705.9	1644.8	1802.3
Instance50-s5-42	793	144	1189.6	1207.3	1132.5	1266.2	1179.8	1347.9
Instance50-s5-43	1644	131	1419.5	1450.1	1315.8	1533.0	1414.8	1631.5
Instance50-s5-44	1016	97	1032.6	1054.6	938.4	1109.3	1045.1	1045.1
Instance50-s5-45	2015	304	1449.9	1470.9	1291.0	1489.5	1435.9	1574.2
Instance50-s5-46	583	194	1069.3	1091.8	917.1	1216.4	1077.7	1107.0
Instance50-s5-47	1417	268	1568.6	1598.6	1468.5	1598.6	1571.5	1740.6
Instance50-s5-48	1062	56	1064.3	1082.1	993.2	1200.9	1082.2	1082.2
Instance50-s5-49	1928	291	1429.1	1434.8	1396.6	1524.6	1404.5	1555.7
Instance50-s5-5	1113	190	1077.2	1101.1	999.2	1172.4	1064.2	1141.7
Instance50-s5-51	1733	128	1384.9	1398.0	1334.9	1431.0	1333.3	1490.9
Instance50-s5-52	644	207	1119.2	1125.6	1020.7	1208.6	1113.4	1128.3
Instance50-s5-53	1856	231	1544.2	1551.5	1481.7	1551.5	1547.3	1715.6
Instance50-s5-54	742	174	1119.4	1133.1	1061.8	1198.7	1117.2	1246.8

Table 5.7: Comparison of BCP, BP and BC-JEP - set 4.

Chapter 6

Final Remarks

We proposed along this thesis models and algorithms for solving routing problems that integrates loading/unloading decisions on warehouses. To be more specific, we deal with the Vehicle Routing Problem with Cross-Docking (VRPCD) and the Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP). To the best of our knowledge, no exact solution approaches were proposed for VRPCD, while only one branch-and-cut algorithm is available for 2E-CVRP.

The VRPCD integrates routing decisions of vehicles with loading/unloading operations at Cross-Docking, a special kind of warehouse without long-term inventory. We introduced three mathematical formulations for the VRPCD. The first is based on network flow, while the two others are set partitioning based models. The network flow formulation is solved using the *Mixed Integer Programming* package from `CPLEX`. In order to solve the set partitioning based models, we implemented two branch-and-price algorithms. Different strategies were proposed for solving pricing problems on branch-and-price algorithms: Dynamic Programming, Branch-and-cut and Heuristic algorithms. The performance of such algorithms vary according to the instances to be solved. In overall, BP algorithms perform better than `CPLEX` solving the network flow model. They are capable of providing more optimality certificates and also to evaluate sharper lower and upper bounds. However, the performance of branch-and-price algorithms worsen when loose constrained instances are considered, because pricing problems become harder to solve.

According to the VRPCD definition, vehicles must collect loads on suppliers, return to Cross-Docking to perform loads changing before to deliver them to their respective customers. We proposed in this thesis to relax such a constraint enforcing a given vehicle to return at Cross-Docking before to deliver the collected loads. Instead, a vehicle may visit a set of suppliers to collect loads and immediately later to deliver

such loads bypassing the Cross-Docking (if convenient). For such a new problem we named the Pickup and Delivery Problem with Cross-Docking (PDPCD). We proposed a mathematical formulation for the PDPCD based on set partitioning and implemented a branch-and-price algorithm to solve it. Computational results showed that costs could be substantially reduced by introducing pickup and delivery routes on the problem, specially for strict constrained instances.

On the 2E-CVRP, a set of level-1 vehicles is in charge of delivering goods from the depot to intermediate warehouses (named *satellites*). On the *satellites* goods are unloaded, consolidated and loaded into level-2 vehicles, which are in charge of delivering them to their respective customers. In order to solve the 2E-CVRP we proposed a set partitioning based model, which comprises an exponential number of variables and constraints. In addition, we also investigate four families of valid inequalities to strengthen such model. Such inequalities are derived from the CVRP literature. Aiming to evaluate solutions for 2E-CVRP using the set partitioning model (including valid inequalities) we implemented a branch-and-cut-and-price algorithm, which outperforms the previous approaches of the literature. It is capable of providing optimality certificates for instances left unsolved by previous exact methods. Moreover, for those instances on which our method is not able to provide optimality certificates, a stronger lower bound could be evaluated and hopefully a good quality upper bound.

As further research steps, we suggest:

- **To investigate valid inequalities for VRPCD and PDPCD:** As described in chapter 5, branch-and-price algorithms may benefit from the introduction of valid inequalities to strengthen their LP lower bounds. Because our algorithms for solving VRPCD and PDPCD do not include any valid inequality, such an issue is a promising research task. To be more specific, valid inequalities derived from the CVRP may also be applied to VRPCD and PDPCD cases, since CVRP lies on the VRPCD and PDPCD structure.

- **To relax formulations for VRPCD and PDPCD aiming to evaluate them faster:** In the 2E-CVRP, we obtain formulation F2 by relaxing F1 to allow routes to visit vertex more than once. Although F2 is weaker than F1, it is faster to evaluate. Because of that, F2 outperforms F1 on solving 2E-CVRP instances. Similar procedures may be done for relaxing the proposed models of VRPCD and PDPCD, aiming to obtain better algorithms. Tree based relaxations are recommended.

- **To include time windows constraints on the VRPCD, PDPCD and 2E-CVRP:** The problems under consideration here neglect time windows constraints on the nodes. Such a constraint may be useful in many routing problems. For that reason, we suggest to consider in next researches the introduction of time windows constraints on such problems. Specially on VRPCD, since most of works on the literature deal with such constraints.

- **To consider another decomposition approaches for solving VRPCD, PDPCD and 2E-CVRP:** we studied in this thesis formulations that comprises an exponential number of variables to model VRPCD, PDPCD and 2E-CVRP. To evaluate lower bounds for such formulations, column generation algorithms must be used due to the large number of variables. Another research step consists of studying different decomposition approaches, as Lagrangean Relaxation, Branch-and-cut or Benders Decomposition, for solving the proposed problems.

- **To implement heuristic approaches to approximate solutions for VRPCD, PDPCD and 2E-CVRP:** we focus on the exact solution approaches for solving the problems under consideration here. However, due to the complexity of such problems, such approaches are capable of solving only medium sized instances. A straightforward approach for evaluating good quality solutions faster consists of implementing heuristic algorithms. Indeed, most of the solution approaches in the literature lies on heuristic algorithms.

Bibliography

Achuthan, N., Caccetta, L., and Hill, S. (2003). An improved branch-and-cut algorithm for the capacitated vehicle routing problem. *Transportation Science*, 37:153–169.

Agarwal, Y., Mathur, K., and Salkin, H. (1989). A set-partitioning based exact algorithm for the vehicle routing problem. *Networks*, 19:731–739.

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2005). TTTplots - a perl program to create time-to-target plots. Technical report, AT&T Labs Research.

Alvarenga, G., Mateus, G., and de Tomi, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers and Operations Research*, 34(6):1561–1584.

Anily, S. (1996). The vehicle-routing problem with delivery and back-haul options. *Naval Research Logistic Quarterly*, 43:415–434.

Apte, U. M. and Viswanathan, S. (2000). Effective cross docking for improving distribution efficiencies. *International Journal of Logistics*, 3:291–302.

Araque, J., Kudva, G., Morin, T., and Pekny, J. (1994). A branch-and-cut algorithm for the vehicle routing problem. *Annals of Operations Research*, 50:37–59.

Araque, J. R., Hall, L. A., and Magnanti, T. L. (1990). Capacitated trees, capacitated routing and associated polyhedra. Core discussion papers, Center for Operations Research and Econometrics, Catholic University of Louvain.

Baldacci, R., Bartolini, E., and Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59:414–426.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115:351–385.

Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A. (2004). An Exact Algorithm for the Capacitated Vehicle Routing Problem based on Two-commodity Network Flow Formulation. *Operations Research*, 52(4):723–738.

Balinkski, M. and Quandt, R. (1964). On an integer program for a delivery problem. *Operations Research*, 12:300–304.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-price: Column Generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. *Journal of the Operations Research Society*, 41:1069–1072.

Bell, W., Dalberto, L., Fisher, M., Greenfield, A., Jaimkumar, R., Kedia, P., Mack, R., and Prutzman, P. (83). Improving the distribution of industrial gases with an online computerized routing and scheduling optimizer. *Interfaces*, 13:4–23.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 15:1–31.

Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers and Operations Research*, 37:32–41.

Boysen, N. and Fliedner, M. (2010). Cross-docking scheduling: Classification, literature review and research agenda. *Omega*, 38:413–422.

Campbell, A. M., Clarke, L. W., and Savelsberg, M. W. P. (2002). *The Vehicle Routing Problem*, chapter Inventory Routing in Practice, page 350. SIAM Monographs on Discrete Mathematics and Apllications.

Campos, V., Corberan, A., and Mota, E. (1991). Polyhedral results for a vehicle routing problem. *European Journal of Operational Research*, 52:75.

Caprara, A. and Fischetti, M. (1997). *Annotated bibliographies in combinatorial optimization*, chapter Branch-and-Cut algorithms, pages 45–63. John Wiley and Sons.

Chen, F. and Lee, C.-Y. (2009). Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research*, 193:59–72.

Chen, Z.-L. and Xu, H. (2006). Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40:74–88.

Chiang, W.-C. and Russell, R. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63:3–27.

Chiang, W.-C. and Russell, R. (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9:417–430.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle dispatching problem. *Operations Research Quartely*, 20:309–318.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem based on the spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282.

Chvátal, V. (1973). Edmonds polytopes and weakly hamiltonian graphs. *Mathematical Programing*, 5:29–40.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581.

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

Cornuéjols, G. and Harche, F. (1993). Polyhedral study of the capacitated vehicle routing problem. *Mathematical Programming*, 60:21–52.

Crainic, T., Mancini, S., Perboli, G., and Tadei, R. (2011). Multi-start heuristics for the two-echelon vehicle routing problem. *Lecture Notes in Computer Science*, 6622:179–190.

Crainic, T., Perboli, G., Mancini, S., and Tadei, R. (2010). Two-echelon vehicle routing problem: A satellite location analysis. *Procedia: Social & Behavioral Sciences*, 2:5944–5955.

Crainic, T., Ricciardi, N., and Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation Science*, 43:432--454.

Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M. (1954). Solution of a Large Scale Traveling Salesman Problem. *Operations Research*, 2:393–410.

Dantzig, G. B. and Ramser, R. H. (1959). The Truck Dispatching Problem. *Management Science*, 6:80–91.

Deif, I. and Bodin, L. (1984). Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with Backhauling. In *Babson College Conference on Software Uses in Transportation and Logistic Management*, pages 75–96.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354.

Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). *Network Routing - Handbooks in Operations Research and Management Science*, chapter Time constrained routing and scheduling, pages 35–139. North-Holland, Amsterdam.

Dondo, R., Méndez, C. A., and Cerdá, J. (2009). Managing distribution in supply chain networks. *Industrial & Engineering Chemistry Research*, 48:9961--9978.

Dondo, R., Méndez, C. A., and Cerdá, J. (2011). The multi-echelon vehicle routing problem with cross docking in supply chain management. *Computers and Chemical Engineering*, 35:3002--3024.

Duhamel, C., Potvin, J.-Y., and Rousseau, J.-M. (1997). A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science*, 31:49–59.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22.

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

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Fischetti, M., Toth, P., and Vigo, D. (1994). A Branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42:846–859.

Fisher, M. and Jaikumar, R. (1981). A generalized assignment heuristic for the vehicle routing problem. *Networks*, 11:109–124.

Fisher, M. L. (1994). Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees. *Operations Research*, 42:626–642.

Forger, G. (1995). Ups starts world's premiere cross-docking operation. *Modern Materials Handling*, pages 36–38.

Fukasawa, R., Longo, H., de Aragão, J. L. M. P., Reis, M., and Uchoa, E. (2006). Robust Branch-and-cut-and-price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3):491–511.

Fumero, F. and Vercellis, C. (1999). Synchronized development of production, inventory, and distribution schedules. *Transportation Science*, 33:330–340.

Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290.

Gendreau, M., Laporte, G., and Potvin, J. Y. (1997). Vehicle routing: Modern Heuristics. In Aarts, E. and Lenstra, J., editors, *Local Search in Combinatorial Optimization*, pages 311–336.

Geoffrion, A. M., Graves, G. W., and Lee, S. J. (1982). A management support system for distribution planning. *INFOR*, 20(4):287–314.

Glover, F. (1997). A template for scatter search and path relinking. *Lecture Notes in Computer Science*, 1363:13–54.

Goemans, M. (1994). The steiner tree polytope and related polyhedra. *Mathematical Programming*, 63:157--182.

Goetschalckx, M. and Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42:39–51.

Goetschalckx, M. and Jacobs-Blecha, C. (1993). The vehicle routing problem with backhauls: Properties and solution algorithms. Technical report MHRC-TR-88-13, Georgia Institute of Technology, Atlanta.

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

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I. M. (1998). Metaheuristics in Vehicle Routing. In Crainic, T. G. and Laporte, G., editors, *Fleet management and logistics*, pages 33–56. Kluwer.

Gouveia, L. (1995). A result on projection for the vehicle routing problem. *European Journal of Operational Research*, 85:610–645.

Grötschel, M. and Padberg, M. W. (1979a). On the symmetric travelling salesman problem i: inequalities. *Mathematical Programming*, 16:265–280.

Grötschel, M. and Padberg, M. W. (1979b). On the symmetric travelling salesman problem ii: lifting theorems and facets. *Mathematical Programming*, 16:281–302.

Gue, K. R. (2001). Cross-docking: Just-in-time for distribution. Teaching Notes-Naval Postgraduate School - Monterey, CA.

Hadjiconstantinou, E., Christofides, N., and Mingozzi, A. (1995). A new exact algorithm from the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Annals of Operations Research*, 61:21–44.

Homberger, J. and Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37:297–318.

Huisman, D., Jans, R., Peeters, M., and Wagelmans, A. P. M. (2005). Column Generation. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Combining Column Generation and Lagrangean Relaxation*, pages 247–270. Springer.

Jepsen, M., Spoorendonk, S., and Ropke, S. (2012). A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*.

Jepsen, M. K., Petersen, B., and Spoorendonk, S. (2008). A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint. Technical report, Datalogistik Institut Kobenhavns Universitet - DIKU.

Jung, J. and Mathur, K. (2009). An efficient heuristic algorithm for a two-echelon joint inventory and routing problem. *Transportation Science*, 41:55–73.

Kallehauge, B., Larsen, J., and Madsen, O. (2006). Lagrangian duality applied to the vehicle routing problem with time windows. *Computers and Operations Research*, 33:1464–1487.

Knight, K. and Hofer, J. (1968). Vehicle scheduling with timed and connected calls: A case study. *Operational Research Quarterly*, 19:299–310.

Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116.

Kohl, N. and Madsen, O. (1997). An optimization algorithm for the vehicle routing problem with time windows based on lagrangean relaxation. *Operations Research*, 45:395–406.

Laporte, G. (1991). The vehicle routing problem: An overview on exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43:408–416.

Laporte, G. and Bourjolly, J. (1984). Some further results on k-star constraints and comb inequalities. Cahiers du gerad, g-84-17, Ecole des Hautes Etudes Commerciales.

Laporte, G. and Nobert, Y. (1984). Comb inequalities for the vehicle routing problem. *Methods of Operations Research*, 51:271–276.

Lee, Y. H., Jung, J. W., and Lee, K. M. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers and Industrial Engineering*, 51(2):247–256.

Lejeune, M. A. and Ruszczynski, A. (2007). An efficient trajectory method for probabilistic production-inventory-distribution problems. *Operations Research*, 55:378–394.

Letchford, A. N., Eglese, R. W., and Lysgaard, J. (2002). Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 94:21–40.

Liao, C.-J., Lin, Y., and Shih, S. C. (2010). Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications*, 37(10):6868 -- 6873.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected Topics in Column Generation. *Operations Research*, 53(6):1007–1023.

Lucena, A. (1992). Steiner problem in graphs: Lagrangean relaxation and cutting planes. *COAL Bulletin*, 21:2--8.

Lysgaard, J. (2004). CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem. http://www.hha.dk/l̃ys/CVRPSEP.htm.

Lysgaard, J., Letchford, A. N., and Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423--445.

Mak, K. L. and Wong, Y. S. (1995). Design of integrated production-inventory-distribution systems using genetic algorithm. In *1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 454–460.

Margot, F., Prodon, A., and Liebling, T. M. (1994). Tree polytope on 2-trees. *Mathematical Programming*, 63:183--192.

Martinhon, C., Lucena, A., and Maculan, N. (2004). Stronger K-Tree Relaxations for the Vehicle Routing Problem. *European Journal of Operational Research*, 158:56–71.

Miller, C., Tucker, A., and Zemlin, R. (1960). Integer programming formulations and traveling salesman problems. *Journal of the ACM*, 7:326–329.

Miller, D. (1995). A matching based exact algorithm for capacitated vehicle routing problems. *ORSA Journal on Computing*, 7:1–9.

Mingozzi, A., Giorgi, S., and Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33:315–329.

Mladenovic, N. and Hansen, P. (1997). A variable neighborhood search. *Computers and Operations Research*, 24:1097–1100.

Moin, N. H. and Salhi, S. (2007). Inventory routing problems: A logistical overview. *The Journal of the Operational Research Society*, 58:1185–1194.

Musa, R., Arnaout, J.-P., and Jung, H. (2010). Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. *Computers and Industrial Engineering*, 59:85 -- 92.

Naddef, D. and Rinaldi, G. (2002). *The Vehicle Routing Problem*, chapter Branch-and-cut algorithms for the capacitated VRP. SIAM Monographs on Discrete Applied Mathematics.

Naddef, D. and Thienel, S. (2002). Efficient separation routines for the symmetric traveling salesman problem i: general tools and comb separation. *Mathematical Programming*, 92:237–255.

Ohlmann, J. W., Fry, M. J., and Thomas, B. W. (2008). Route Design for Lean Production Systems. *TRANSPORTATION SCIENCE*, 42(3):352–370.

Osman, I. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451.

Padberg, M. W. and Rao, M. R. (1982). Odd minimum cut-sets and b-matchings. *Mathematics of Operational Research*, 7:67–80.

Padberg, M. W. and Rinaldi, G. (1990). Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257.

Perboli, G., Pezzella, F., and Tadei, R. (2008a). Eve-opt: a hybrid algorithm for the capacitated vehicle routing problem. *Mathematical Methods of Operations Research*, 68:361–382.

Perboli, G. and Tadei, R. (2010). New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic Notes on Discrete Mathematics*, 36:639–646.

Perboli, G., Tadei, R., and Vigo, D. (2008b). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. Technical report 55, CIRRELT.

Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45:364–380.

Potvin, J.-Y. and Bengio, S. (1996). The vehicle routing problem with time windows - part ii: Genetic search. *INFORMS Journal on Computing*, 8:165–172.

Potvin, J.-Y., Kervahut, T., Garcia, B., and Rousseau, J.-M. (1996). The vehicle routing problem with time windows. part i: Tabu search. *INFORMS Journal on Computing*, 8:158–164.

Potvin, J.-Y. and Rousseau, J.-M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66:331–340.

Pullen, H. and Webb, M. (1967). A computer application to a transport scheduling problem. *Computer Journal*, 10:10–13.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., and Trotter, L. E. (2002). On the capacitated vehicle routing problem. *Mathematical Programming Serie B*, 94:343–359.

Ricciardi, N., Tadei, R., and Grosso, A. (2002). Optimal facility location with random throughput costs. *Computers and Operations Research*, 29:593–607.

Ropke, S. and Cordeau, J.-F. (2006). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43:267–286.

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

Salani, M. (2005). *Branch-and-price algorithms for vehicle routing problems*. PhD thesis, Universita degli studi di Milano.

Santos, F. A., Cunha, A. S., and Mateus, G. R. (2010). Modelos de otimização para o problema de roteamento de veículos com cross-docking (in portuguese). In *Simpósio Brasileiro de Pesquisa Operacional - SBPO*.

Santos, F. A., Cunha, A. S., and Mateus, G. R. (2011a). Um algoritmo branch-and-price para o problema de roteamento de veículos com *Cross-Docking* para frotas heterogêneas (in portuguese). In *Simpósio Brasileiro de Pesquisa Operacional - SBPO*.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2011b). A Novel Column Generation Algorithm for the Vehicle Routing Problem with Cross-Docking. In *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 412--25. Springer.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2011c). A branch-and-price algorithm for a vehicle routing problem with cross-docking. In *LAGOS'11 - VI Latin-American Algorithms, Graphs and Optimization Symposium*, volume 37 of *Electronic Notes in Discrete Mathematics*, pages 249--54. Elsevier.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2012a). Branch-and-price algorithms for the two-echelon vehicle routing problem. *Optimization Letters*. DOI: 10.1007/s11590-012-0568-3.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2012b). The pickup and delivery problem with cross-docking. *Computers and Operations Research*. DOI: 10.1016/j.cor.2012.11.021.

Savelsbergh, M. (1985). Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305.

Savelsbergh, M. W. P. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29:17–29.

Solomon, M. (1986). On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks*, 16:161–174.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265.

Stein, D. M. (1978a). An asymptotic, probabilistic analysis of a routing problem. *Mathematics of Operations Research*, 3:89–101.

Stein, D. M. (1978b). Scheduling dial-a-ride transportation systems. *Transportation Science*, 12:232–249.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186.

Tarantilis, C. (2012). Adaptive multi-restart tabu search algorithm for the vehicle routing problem with cross-docking. *Optimization Letters*. DOI: 10.1007/s11590-012-0558-5.

Thangiah, S., Potvin, J.-Y., and Sun, T. (1996). Heuristic approaches to vehicle routing with backhauls and time windows. *Computers and Operations Research*, 23:1043–1057.

Topan, E., Bayindir, Z. P., and Tan, T. (2009). An exact solution procedure for multi-item two-echelon spare parts inventory control with batch ordering in the central warehouse. *Operations Research Letters*, 38:454–461.

Toth, P. and Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31:372–385.

Toth, P. and Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113:528–543.

Toth, P. and Vigo, D. (2002a). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123:487–512.

Toth, P. and Vigo, D., editors (2002b). *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM.

Vanderbeck, F. and Wolsey, L. A. (1996). An exact algorithm for IP column generation. *Operations Research Letters*, 19(4):151 – 159.

Wen, M., Larsen, J., Clausen, J., Cordeau, J.-F., and Laporte, G. (2009). Vehicle routing with cross-docking. *Journal of Operational Research Society*, 60:1708–1718.

Wilson, H., Sussman, J., Wang, H., and Higonnet, B. (1971). Scheduling algorithms for dial-a-ride systems. Technical report USL TR-70-13, Urban Systems Laboratory, MIT.

Witt, C. E. (1998). Crossdocking: Concepts demand choice. *Material Handling Engineering*, 53:44--49.

Yu, W. and Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184(1):377–396.

Yung, K.-L., Tang, J., Ip, A. W. H., and D.Wang (2006). Heuristics for joint decisions in production, transportation, and order quantity. *Transportation Science*, 40:99–116.