

Two-echelon vehicle routing problems in city logistics : approaches based on exact methods of mathematical optimization

Guillaume Marques

▶ **To cite this version:**

Guillaume Marques. Two-echelon vehicle routing problems in city logistics : approaches based on exact methods of mathematical optimization. Automatic Control Engineering. Université de Bordeaux, 2020. English. NNT : 2020BORD0199 . tel-03097659

HAL Id: tel-03097659
https://theses.hal.science/tel-03097659

Submitted on 5 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE PHYSIQUE ET SCIENCES DE L'INGÉNIEUR
AUTOMATIQUE, PRODUCTIQUE, SIGNAL ET IMAGE, INGÉNIERIE COGNITIQUE

Par Guillaume MARQUES

PROBLÈMES DE TOURNÉES DE VÉHICULES SUR DEUX NIVEAUX POUR LA LOGISTIQUE URBAINE

Approches basées sur les méthodes exactes de l'optimisation mathématique

Sous la direction de Rémy DUPAS et Ruslan SADYKOV

Soutenue le 26 Novembre 2020

Membres du jury :

M. CLAUTIAUX, François	Professeur, Université de Bordeaux	Président
Mme. ARCHETTI, Claudia	Professeur associé, ESSEC Business School	Rapporteur
M. LEHUÉDÉ, Fabien	Professeur, IMT Atlantique	Rapporteur
M. TANIGUCHI, Eiichi	Professeur émérite, Kyoto University	Examinateur
Mme. ULRICH NGUEVEU, Sandra	Maître de conférences, INP Toulouse	Examinateur
M. DUPAS, Rémy	Professeur, Université de Bordeaux	Directeur
M. SADYKOV, Ruslan	Chargé de recherche, Inria	Directeur
M. DESCHAMPS, Jean-Christophe	Maître de conférences, Université de Bordeaux	Encadrant
M. VANDERBECK, François	Directeur général, Atoptima	Invité

Abstract

Cette thèse s'intéresse principalement au développement de méthodes d'optimisation mathématique exactes pour résoudre des problèmes de tournées de véhicules dans un réseau de distribution à deux niveaux. Dans un tel réseau, des camions circulent au premier niveau et transportent des marchandises d'un centre de distribution vers des dépôts intermédiaires appelés « satellites ». Au second niveau, des véhicules légers récupèrent les marchandises aux satellites puis livrent les clients. Chaque client doit être visité une seule fois. L'objectif du problème de tournées de véhicules sur deux niveaux est de minimiser le coût total de transport dans un tel réseau de distribution.

Le premier chapitre présente succinctement l'algorithme de « branch-and-cut-and-price » que nous utilisons tout au long de la thèse.

Le second chapitre s'intéresse au « Two-Echelon Capacitated Vehicle Routing Problem ». Nous présentons une nouvelle formulation du problème basée sur des routes qui ne contient pas de variable pour determiner les quantités de marchandises livrées aux satellites. Nous proposons une nouvelle stratégie de branchement qui permet de significativement réduire la taille de l'arbre de branch-and-bound. Enfin et surtout, nous présentons une nouvelle famille d'inégalités valides nommée « satellite supply inequalities ». Nous montrons que cette nouvelle famille améliore la qualité de la borne duale au noeud racine de l'arbre de « branch-and-bound ». Les expérimentations montrent que notre algorithme résout toutes les instances de la littérature qui contiennent jusqu'à 200 clients et 10 satellites. C'est le double de la taille des instances qui étaient résolues à l'optimalité jusqu'ici.

La troisième chapitre s'intéresse au « Two-Echelon Vehicle Routing Problem with Time Windows ». Ici, nous considérons une variante avec des contraintes de précédence aux satellites : un camion doit livrer les marchandises à un satellite avant qu'elles soient chargées dans un véhicule léger. C'est une relaxation de la variante avec synchronisation exacte considérée dans la littérature. Nous traitons les variantes « single trip » et « multi trip » du problème avec contraintes de précédence. Dans la seconde variante, les véhicules légers partent d'un dépôt, récupèrent les marchandises aux satellites, puis effectuent plusieurs tournées. Nous proposons une formulation basée sur les routes dont

le nombre de contraintes de précédence croît exponentiellement en fonction du nombre de satellites. Un algorithme basé sur une coupe minimum est proposé pour séparer ces contraintes. Nous montrons aussi comment prendre en compte ces contraintes dans le problème de pricing de l'algorithme de génération de colonnes. Les expérimentations montrent que notre algorithme peut résoudre à l'optimalité des instances qui contiennent jusqu'à 100 clients. De plus, nous montrons que la variante du problème avec contraintes de précédence donne des résultats identiques à ceux de la variante avec synchronisation exacte pour les instances « single trip » de la littérature.

La quatrième chapitre s'intéresse à des problèmes de tournées de véhicules contenant des contraintes de type sac-à-dos. Nous présentons des coupes de type « route load knapsack ». Ces coupes sont utilisées pour résoudre les trois problèmes suivants: « Capacitated Vehicle Routing Problem with Capacitated Multiple Depots », « Location-Routing Problem » et « Vehicle Routing Problem with Time Windows and Shifts ». Ces problèmes apparaissent lorsque les routes au premier niveau du réseau de distribution à deux niveaux sont fixées. Nos expérimentations permettent de trouver de nouvelles solutions optimales.

Mot-clefs : Optimisation exacte, Génération de coupes et de colonnes, Inégalités valides, Tournées de véhicules avec transbordement, Problèmes de tournées de véhicules à deux niveaux.

Abstract

The main focus of this thesis is to develop mathematical optimization based exact methods to solve vehicle routing problems in two-level distribution systems. In such a system, the first level involves trucks that ships goods from a distribution center to intermediate depots called satellites. The second level involves city freighters that are loaded with goods at satellites and deliver the customers. Each customer must be visited once. The two-echelon vehicle routing problem seeks to minimize the total transportation cost in such a distribution system.

The first chapter gives an overview of the branch-and-cut-and-price framework that we use throughout the thesis.

The second chapter tackles the Two-Echelon Capacitated Vehicle Routing Problem. We introduce a new route based formulation for the problem which does not use variables to determine product flows in satellites. We propose a new branching strategy which significantly decreases the size of the branch-and-bound tree. Most importantly, we suggest a new family of satellite supply inequalities, and we empirically show that it improves the quality of the dual bound at the root node of the branch-and-bound tree. Experiments reveal that our algorithm can solve all literature instances with up to 200 customers and 10 satellites. Thus, we double the size of instances which can be solved to optimality.

The third chapter tackles the Two-Echelon Vehicle Routing Problem with Time Windows. We consider the variant with precedence constraints at the satellites: products should be delivered by an urban truck to a satellite before loading them to a city freighter. This is a relaxation of the synchronisation variant usually considered in the literature. We consider single-trip and multi-trip variants of this problem. In the first one, city freighters start from satellites and do a single trip. In the second one, city freighters start from a depot, load product at satellites, and do several trips. We introduce a route based formulation that involves an exponential number of constraints to ensure precedence relations. A minimum-cut based algorithm is proposed to separate these constraints. We also show how these constraints can be taken into account in the pricing problem of the column generation approach. Experiments show that our algorithm can solve to

optimality instances with up to 100 customers. The algorithm outperforms significantly another recent approach proposed the literature for the single-trip variant of the problem. Moreover, the "precedence relaxation" is exact for single-trip instances.

The fourth chapter considers vehicle routing problems with knapsack-type constraints in the master problem. For these problems, we introduce new route load knapsack cuts and separation routines for them. We use these cuts to solve to optimality three problems: the Capacitated Vehicle Routing Problem with Capacitated Multiple Depots, the standard Location-Routing Problem, and the Vehicle Routing Problem with Time Windows and Shifts. These problems arise when routes at first level of the two-level distribution system are fixed. Our experiments reveal computational advantage of our algorithms over ones from the literature.

Keywords: Exact optimization, Column and cut generation, Valid inequalities, Vehicle routing with transshipments, Two-level vehicle routing problems.

Acknowledgements

I would like to thank Linda, my friends, and my family.

I would like to thank François Vanderbeck for all the research and job opportunities since my master internship.

I would like to thank Rémy Dupas, Ruslan Sadykov, and Jean-Christophe Deschamps for the supervision of my thesis.

I would like to thank Eduardo Uchoa who welcomed me three months at Universidade Federal Fluminense (Niterói, Brazil). This stay was funded by Idex Bordeaux. I would also like to thank all the LOGIS team who has been very nice to me. I thank Anand Subramanian and his team who welcomed me for a week at Universidade Federal da Paraíba (João Pessoa, Brazil) and Thibaut Vidal who invited me to spend one day at Pontifícia Universidade Católica do Rio de Janeiro.

I would like to thank the members of the jury for the time spent reading my thesis and all my colleagues for the nice working environment.

Table of contents

List of figures xiii

List of tables xv

Introduction 1

1 Getting started 5
- 1.1 Linear programming . . . 5
- 1.2 Integer programming . . . 8
- 1.3 General structure of the branch-cut-and-price . . . 14
  - 1.3.1 Cut generation . . . 15
  - 1.3.2 Column generation . . . 16
  - 1.3.3 Generic BCP enhancements . . . 25
  - 1.3.4 Overview of the algorithm . . . 27

2 Two-Echelon Vehicle Routing Problem 31
- 2.1 Literature review . . . 33
- 2.2 Formulation . . . 35
  - 2.2.1 Standard formulation . . . 36
  - 2.2.2 Modified formulation . . . 37
  - 2.2.3 Valid inequalities . . . 39
- 2.3 New family of valid inequalities . . . 40
  - 2.3.1 Satellite supply inequalities . . . 40
  - 2.3.2 Separation of Satellite supply inequalities . . . 42
- 2.4 Branch-cut-and-price algorithm . . . 45
  - 2.4.1 Pricing problem . . . 45
  - 2.4.2 Column and cut generation . . . 46
  - 2.4.3 Branching . . . 46
  - 2.4.4 Primal heuristic . . . 47

2.5 Computational results . . . 47
2.5.1 Instances . . . 48
2.5.2 Experimental analysis of BCP variants . . . 48
2.5.3 Comparison with the state-of-the-art algorithm . . . 50
2.5.4 Experimental results for new instances . . . 51

3 Two-Echelon Vehicle Routing Problem with Time-Windows 53
3.1 Literature review . . . 55
3.2 Problem definition and formulation . . . 57
3.2.1 Standard formulation . . . 60
3.2.2 Modified formulation . . . 61
3.2.3 Valid inequalities . . . 64
3.2.4 Multi-trip variant . . . 65
3.3 Branch-cut-and-price algorithm . . . 66
3.3.1 Pricing problems . . . 67
3.3.2 TLPC separation algorithm . . . 71
3.3.3 The overall algorithm . . . 73
3.3.4 Post-processing . . . 73
3.4 Computational results . . . 75
3.4.1 Literature Instances . . . 76
3.4.2 Results for literature instances . . . 77
3.4.3 Results for new single-trip instances . . . 80
3.4.4 Results for smaller multi-trip instances . . . 80
3.4.5 Results for instances with modified vehicle capacity . . . 81

4 Location-Routing and Related Problems 83
4.1 Literature review . . . 84
4.2 Problem definition and formulation . . . 88
4.2.1 Formulation . . . 88
4.2.2 Robust valid inequalities . . . 90
4.2.3 Route load knapsack cuts . . . 93
4.2.4 Separation of Route load knapsack cuts . . . 95
4.3 Pricing routes . . . 97
4.4 Computational experiments . . . 99
4.4.1 Location-routing instances . . . 100
4.4.2 CVRP-CMD instances . . . 103
4.4.3 VRPTW with Shifts instances . . . 105

Conclusion and Perspectives 109

Bibliography 115

List of figures

1.1 Generic representation of a linear program 6
1.2 Geometric representation of an integer program 9
1.3 Example of a branching constraint 13
1.4 Buckets for a vertex of the bucket graph 19
1.5 Example of a solution to the RCESPP for the CVRP 21
1.6 Example of the tracking of the resource consumption of a lm-R1C 23
1.7 Examples of cycles allowed or forbidden in a priced *ng*-route. 24
1.8 Overview of the column-and-cut generation algorithm 29

2.1 Illustration for the proof of the equivalence of two formulations for the 2E-CVRP 38
2.2 Example of a Satellite supply inequality 41
2.3 Illustration of the linearization of a Satellite supply inequality 42
2.4 Example of a graph to separate Satellite supply inequalities 43

3.1 Examples of transfers at a satellite 58
3.2 Example of a solution that violates a Two-level precedence constraint . . 62
3.3 Illustration for the proof of the equivalence between two formulations for the 2E-VRPTW 64
3.4 Example of extended graph to price second-level single-trip routes 69
3.5 Examples of extended graphs to price second-level multi-trip routes . . . 70
3.6 Instance of a graph to separate Two-level precedence cuts 72
3.7 Overview of the results for multi-trip instances with 25, 50, and 75 customers 81

4.1 Performance profiles of BCP variants tested against Prodhon instances . 102
4.2 Performance profile for BCP variants tested against the CVRP-CMD instances 105
4.3 Performance profile for BCP variants tested against the VRPTW-S instances 106

List of tables

2.1 Sets of 2E-CVRP instances from the literature used for experiments . . . 48
2.2 Comparison of variants of the BCP algorithm 49
2.3 Comparison of two BCP variants with the state-of-the-art exact algorithm for the 2E-CVRP 50
2.4 Improved best-known solutions for the 2E-CVRP literature instances . . 51

3.1 Sets of 2E-VRPTW instances from the literature used for experiments . . 76
3.2 Comparison of our 2E-VRPTW single-trip results with literature 78
3.3 Overview of results for 2E-VRPTW multi-trip instances 79
3.4 Overview of experiments on multi-trip instances 79
3.5 Overview of experiments on new 2E-VRPTW single-trip instances 80
3.6 Overview of results for 2E-VRPTW instances with original vehicle capacity 82
3.7 Overview of results for 2E-VRPTW instances with modified vehicle capacity 82

4.1 Hint on the validity of lifted Depot capacity cuts for the LRP 91
4.2 Comparison of variants of the BCP algorithm on LRP instances `PPW06` . 101
4.3 Comparison of our result on LRP with literature 102
4.4 Performance of BCP_{best} on instances in the set `SL19` 103
4.5 Comparison of variants of the BCP algorithm on CVRP-CMD instances . 104
4.6 Comparison of variants of the BCP algorithm on VRPTW-S instances . . 106

Introduction

This thesis evaluates the performance of exact methods of mathematical optimization to optimize vehicle routing problems in two-level distribution systems. The problem considered in this thesis arises from city logistics whose fundamental concepts are the consolidation of loads and the coordination of operations. The practical purpose of the problem is to deliver customers located in city centers with low capacitated vehicles which deteriorate as little as possible the living conditions of urban dwellers.

As noticed by Crainic (2008), freight transportation competes with people transportation for the capacity of the streets, contributes to congestion, and participates in environmental nuisances such as noise and pollution. In the urban area of Bordeaux, a technical report from Agence d'urbanisme Bordeaux Aquitaine (2019) corroborates these statements. They note that 40% of freight movements take place between 7 am and 10 am which are also the rush hours of car traffic. Moreover, they note that about 25% of pickup and delivery operations are performed by freight vehicles illegally parked. However, the proportion of trucks with a capacity greater than 3.5 tons decreases in the urban area. In 1994, half of the movements were performed by trucks. In 2013, it was less than a third.

Over the years, city centers are becoming less accessible to vans and trucks whereas electronic commerce grows and urban migration increases. Traffic restrictions in city centers are motivated for different reasons. One reason is the architecture of the city. Some streets may be too narrow to accommodate trucks. Other reasons are political and their overall aim is to improve the living conditions of urban dwellers, especially improve air quality. In that case, all types of motor vehicles may face restrictions. For instance, the Paris city council closed to traffic a large part of the riverside road in 2016. It was an important arterial road that crossed Paris from east to west and it is now dedicated to pedestrians and buses. In Barcelona, they set up small neighbourhoods with restricted access to vehicles. Vehicles drive outside the neighbourhoods. Inside the neighbourhoods, traffic has given way to living places for residents with parks, playgrounds, and bicycle lanes.

In France, this trend is very clear after the lockdown put in place in March 2020 to limit the spread of the coronavirus (COVID-19). Indeed, the Government of the French Republic implemented the « Plan vélo » to prevent people from abandoning public transportation in favor of private cars. This plan has notably resulted in the creation of numerous cycle paths in major cities. In the urban area of Bordeaux, for instance, a few streets have become one-way streets and the capacity of several arterial roads has been reduced to create lanes dedicated to buses and cycles. Similar developments take place in several European cities as mentioned by some newspapers.

These trends bring us to consider a distribution system in which trucks are banned from the city centers. To this end, trucks ship freight from depots to intermediate depots, called satellites, located within the urban areas. Then, smaller vehicles, called city freighters, collect freight at satellites and deliver the customers in the city center. City freighters can be cargo bikes or electric delivery vans for instance. This way of transporting goods is called a two-echelon distribution system. Urban trucks operate at the first level and city freighters at the second level. Transportation systems require planning at the strategic level (design of the system), tactical level (vehicles routing), and operational level (staff work schedules). In this thesis, we focus on the tactical level. This results in the Two-Echelon Vehicle Routing problem.

From a mathematical optimization perspective, this problem is very challenging. In standard vehicle routing problems, routes are linked by a polynomial number of constraints. In the case of the two-echelon distribution system, since the routes taken by the city freighters fully depends on the routes taken by urban trucks, we will see that it leads to an exponential number of constraints. So, there is a strong dependency between the routes. Moreover, exact algorithms are useful as they are generally used in practice to obtain lower bounds on the value of an optimum solution to the problem. We can then use these bounds to evaluate the quality of heuristics.

The thesis is organized as follows. In the first chapter, we recall the main concepts of mathematical programming and we give an overview of the branch-and-cut-and-price framework used throughout the thesis. This framework embeds several state-of-the-art components to solve efficiently vehicle routing problems.

In the second chapter, we tackle the Two-Echelon Capacitated Vehicle Routing Problem. We review the literature. We introduce a new route based formulation for the problem which does not involve variables to determine freight flows in satellites. We propose a new branching strategy on the number of trucks visiting satellites which significantly decreases the size of the branch-and-bound tree. Most importantly, we suggest a new family of valid inequalities, called satellite supply inequalities, and we

empirically show that it improves the quality of the dual bound at the root node of the branch-and-bound tree. Experiments reveal that our algorithm can solve all literature instances with up to 200 customers and 10 satellites. Thus, we double the size of instances that can be solved to optimality. This work has been presented in ROADEF 2019, Verolog 2019, and the Autumn school on Advanced BCP Tools: VRPSolver and Coluna . It has been published in Marques et al. (2020).

The third chapter tackles the Two-Echelon Vehicle Routing Problem with Time Windows. We consider the variant with precedence constraints at the satellites: products should be delivered by an urban truck to a satellite before loading them to a city freighter. This is a relaxation of the synchronization variant usually considered in the literature. We consider single-trip and multi-trip variants of the problem with precedence constraints. In the first variant, city freighters start from satellites and do a single trip. In the second one, city freighters start from a depot, load product at satellites, and do several trips. We introduce a route based formulation that involves an exponential number of constraints to ensure precedence relations. A minimum-cut based algorithm is proposed to separate these constraints. We also show how these constraints can be taken into account in the pricing problem of the column generation approach. Experiments show that our algorithm can solve to optimality instances with up to 100 customers for both single-trip and multi-trip variants. The algorithm outperforms significantly another recent approach proposed in the literature for the single-trip variant of the problem. At last, we show that the "precedence relaxation" is very tight for the latter instances: all solutions can be transformed into solutions with exact synchronization with the same cost. A part of this work has been presented in ROADEF 2020. We will submit an article to Transportation Science.

The fourth chapter considers mainly the location routing problem which has two levels of decisions. First, we have to choose which depot will be open (location), and second, we plan routes from the opened depots to deliver the customers (routing). It is part of the work I did during my stay at Universidade Federale de Fluminense. In this chapter, we consider vehicle routing problems with knapsack-type constraints in the master problem. For these problems, we introduce a new family of valid inequalities named route load knapsack cuts. We also propose a separation algorithm for these cuts. We use these cuts to solve to optimality three problems: the Capacitated Vehicle Routing Problem with Capacitated Multiple Depots, the standard Location-Routing Problem, and the Vehicle Routing Problem with Time Windows and Shifts. These problems are subproblems of two-echelon vehicle routing problems because they arise when the solution

to the first level is fixed. Our experiments reveal the computational advantage of our algorithms over ones from the literature. An article will be submitted.

Chapter 1

Getting started

In this chapter, we recall the main concepts of mixed-integer linear programming theory. These are the foundations of the algorithmic framework used throughout the thesis. We then give an overview of the framework and describe its key features.

1.1 Linear programming

Mathematical optimization seeks to reach an optimal solution among a set of solutions to a problem. The problem is usually modelled as a mathematical program, introduced in Kantorovich (1939) to model an economic allocation problem, that contains constraints defining the set of feasible solutions and an objective function associating each solution to a value telling how "good" is this solution. In the case of linear optimization, such a program has the form :

$$[P1] \equiv \text{ minimize} \quad c^\intercal x \tag{1.1}$$

$$\text{subject to} \quad Ax \geq b \tag{1.2}$$

$$x \geq 0 \tag{1.3}$$

where x is a vector of m real variables. Vector of costs $c \in \mathbb{R}^m$, coefficient matrix $A \in \mathbb{R}^{m \times n}$, and right-hand side vector $b \in \mathbb{R}^n$ are known. Expression (1.1) is the objective function, expression (1.2) defines a collection of n constraints, and expression (1.3) defines the domains of variables. A solution with the minimum objective value is called an optimal solution.

From the geometric perspective, each constraint of the linear program is a half-space. The intersection of these half-spaces forms a convex polyhedron which is the set of

feasible solutions. The objective function is a hyperplane. In the case of minimization problem, we get an optimal solution by sliding down the latter hyperplane while keeping it intersected with the polyhedron. Figure 1.1 illustrates such an interpretation and we clearly see that, in this example, there is a single optimal solution which is a vertex of the polyhedron.

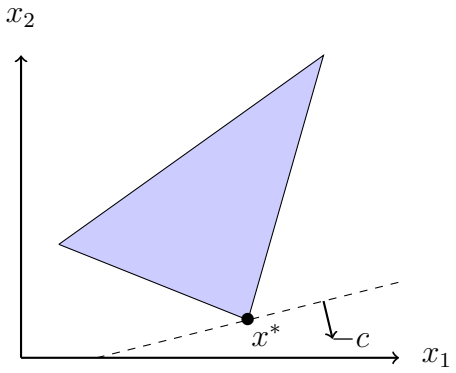


Figure 1.1: Geometric representation of a linear program. It minimizes objective function c. The optimal solution is x^*.

As mentioned in Dantzig (1990), Dantzig proposed the simplex method to optimize linear programs in 1947. This method is based on the geometric interpretation of a linear program. Basically, the simplex method starts from a vertex of the polyhedron and moves along an incident edge chosen by *pivot rules*. A pivot rule can consist, for instance, in choosing an incident edge that goes to a vertex that improves the value of the objective function. The algorithm stops when it finds a vertex corresponding to an optimal solution. This algorithm has an exponential worst-case complexity but it is efficient in practice. There is another class of efficient algorithms known as interior point methods and proposed in Karmarkar (1984). These algorithms have polynomial worst-case complexity for linear optimization. We will not go into details because we do not use these methods in our work. However, it is worth mentioning them because their complexity shows that a linear program may be solved in polynomial time.

A central concept in mathematical optimization is *duality*. An optimization problem can be viewed from two perspectives: the *primal problem* and the *dual problem*. The problem as originally formulated is usually called the primal problem. Here, problem [P1] is the primal problem. The dual problem can be derived from the primal one by applying a method called Lagrangian relaxation.

The *Lagrangian relaxation* is a way to handle the constraints by penalizing their violation in the objective function. Let $y \geq 0$ be a vector of Lagrangian multipliers that contains one entry for each constraint. The *Lagrangian function* of problem [P1]

$$\mathcal{L}(x, y) = c^\intercal x + y^\intercal(b - Ax) = y^\intercal b + (c - y^\intercal A)x \tag{1.4}$$

yields a lower bound on the value of feasible solutions to [P1]. The *dual Lagrangian problem*

$$g(y) = \min_{x \geq 0} \mathcal{L}(x, y) = y^\intercal b + \min_{x \geq 0}\{(c - y^\intercal A)x\} \tag{1.5}$$

looks for the best solution for a given vector of Lagrangian multipliers. Given vector $y \geq 0$, entries of vector $c - y^\intercal A$ are the *reduced costs* of variables x. The reduced cost of a variable defines a lower bound on how much the objective function increases when the value of the variable is incremented by one. Therefore, in the case of a minimization problem, variables with negative reduced costs may improve the objective function. The value of $g(y)$ yields a lower bound on the value of the optimal solution to [P1] and to get the best lower bound, we solve

$$[RP] \equiv \max_{y \geq 0} g(y) \equiv \max_{y \geq 0}\{y^\intercal b + \min_{x \geq 0}\{(c - y^\intercal A)x\}\} \tag{1.6}$$

We note that if the coefficient of x in (1.6) is non-negative, then the value of lower bound $g(y)$ is $y^\intercal b$; $-\infty$ otherwise. From that and because an infinite lower bound is uninformative, we obtain the following problem :

$$[DP1] \equiv \text{ maximize} \quad y^\intercal b \tag{1.7}$$

$$\text{subject to} \quad y^\intercal A \leq c \tag{1.8}$$

$$y \geq 0 \tag{1.9}$$

Problem $[DP1]$ is called the *dual problem* of $[P1]$ and y are called *dual variables*. The value of a dual variable, also called *dual price*, is a lower bound on how much the objective function increases each time the right-hand side of the associated constraint is incremented by one. Any feasible solution to $[DP1]$ gives a lower bound, also called *dual bound*, on the value of the optimal solution to $[P1]$. Conversely, $[P1]$ is the *primal problem* and the value of any feasible solution to $[P1]$ is called a *primal bound*.

A key point about linear optimization and duality is that, given x^* optimal solution of $[P1]$ and y^* optimal solution of $[DP1]$, $c^\intercal x^* = b^\intercal y^*$ holds. This is called *strong duality*.

1.2 Integer programming

Linear programming offers a generic way to model optimization problems in which the decisions consist of determining amounts of uncountable things. However, we often face problems that involve choices or indivisible things. For example, a solution stating that a company needs 2.65 vehicles and 3.12 employees to deliver its customers cannot be considered as feasible. Such problems are *combinatorial problems* and can be written as

$$\min\{c(s) \mid s \in S\}$$

where c is the objective function and S is the set of feasible solutions which is often so large that an exhaustive enumeration of solutions would take ages (combinatorial explosion). A combinatorial problem has a valid formulation if we can describe a polyhedron $P = \{x \mid Ax \geq b\}$ such as

$$\min\{c^\intercal x \mid x \in P \cap \mathbb{N}^m\} \equiv \min\{c(s) \mid s \in S\}. \tag{1.10}$$

We can rewrite this problem as an *integer linear program*, that is a linear program in which variables take discrete values :

$$[P2] \equiv \min\{c^\intercal x \mid Ax \geq b,\ x \in \mathbb{Z}_+^m\}. \tag{1.11}$$

Since an exhaustive enumeration of the solutions is not always possible, we need to enumerate them in a smart way. To this end, we use the concept of *relaxation* which consists in relaxing some constraints to make the problem easier to solve. We use three types of relaxations :

- *Lagrangian relaxation* (presented in the previous section) that consists in penalizing the violation of some constraints in the objective function,
- *linear relaxation* that consists in ignoring the integrality constraints of the variables,
- *combinatorial relaxation* that consists in removing difficult constraints from the formulation.

Relaxing a formulation leads to an easier problem that provides a dual bound on the optimal value. This dual bound is a useful piece of information to explore the space of solutions in an intelligent way.

From the geometric point of view, feasible solutions to an integer program are integer points inside the polyhedron as we can see in the left picture of figure 1.2. We note that an optimal solution is not necessarily a vertex of the polyhedron and therefore linear programming algorithms applied to the linear relaxation do not necessarily find a feasible solution to the problem. However, integer optimization methods are built on top of linear optimization methods. Therefore, the ideal formulation is the one whose polyhedron has integer points as vertices. Given $(x^s)_{s\in S}$ the set of integer points that correspond to the feasible solutions to S, the ideal formulation is thus the polyhedron described by the convex combination of these feasible solutions

$$\text{conv}(S) = \left\{ \sum_{s\in S} \lambda_s x^s \mid \sum_{s\in S} \lambda_s = 1, \lambda_s \geq 0 \; s \in S \right\} \tag{1.12}$$

also known as the convex hull of the feasible solution in S. Indeed, the optimal solution to the linear relaxation of the ideal formulation is the optimal solution to the problem. Right picture of figure 1.2 gives an example of ideal formulation. Unfortunately, there is no efficient procedure to find the ideal formulation but we know ways to improve the quality of the formulation, *i.e.* to make the formulation stronger, and so improve the dual bound provided by the linear relaxation. Consequently, we look for a formulation whose linear relaxation is as tight as possible around the convex hull of the feasible solutions.

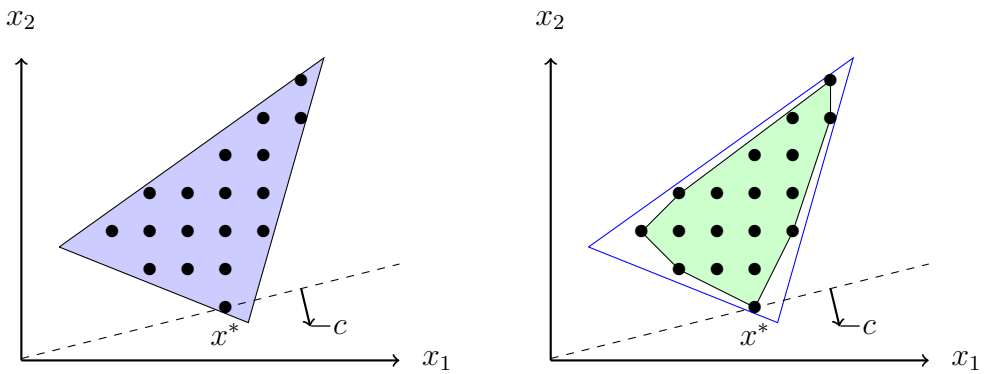


Figure 1.2: Geometric representation of an integer program, its feasible solutions (black dots), its linear relaxation (blue area), and its ideal formulation (green area).

A strong formulation may involve an exponential number of constraints. These constraints must be incrementally added in the formulation to avoid making the linear relaxation intractable. This procedure is named *cut generation* and works as follows. First, it solves (the linear relaxation of) the formulation. Second, it calls a *separation oracle* that returns a certain number of constraints violated by the solution to (the linear relaxation of) the formulation. Third, it adds the violated constraints to the formulation and returns to the first step. Cut separation stops once the separation oracle does not find any violated constraints or when the new constraints barely improve the dual bound. The latter condition is also known as *tailing-off condition*.

There are two types of constraints. The first type of constraints includes essential ones. They are inherent in the definition of feasible solutions. The separation oracle must be an exact algorithm that returns at least one violated constraint if it exists. Indeed, if the algorithm fails to separate some essential constraints, the final solution may be infeasible. The second type of constraints includes valid inequalities. Such constraints strengthen the linear relaxation of the formulation without changing the set of integer feasible solutions. In this case, the separation oracle is either an exact algorithm or a heuristic that seeks to find violated inequalities in a reasonable time.

Example 1. *(Example of valid inequalities - Chvátal-Gomory procedure).*

Consider $P = \{\sum_{j\in J} a_{ij}x_j \leq b_i \ i \in I, \ x \in \mathbb{Z}_+^{|J|}\}$ *with variables* $(x_j)_{j\in J}$*, and real constants* $(a_{ij})_{i\in I, j\in J}$ *and* $(b_i)_{i\in I}$*. The Chvátal-Gomory procedure uses the fact that it is true that, for* $i \in I$*,* $\sum_{j\in J} a_{ij}x_j \leq b_i \implies \sum_{j\in J}\lfloor a_{ij}\rfloor x_j \leq b_i \implies \sum_{j\in J}\lfloor a_{ij}\rfloor x_j \leq \lfloor b_i\rfloor$*.*

Moreover, given a vector of multipliers $\alpha \in \mathbb{R}_+^{|I|}$*, one entry associated to each constraint, the combination* $\sum_{j\in J}\sum_{i\in I}\alpha_i a_{ij}x_j \leq \sum_{i\in I}\alpha_i b_i$ *leads to a valid inequality for* P*. Thus, the following inequality is also valid for* P*.*

$$\sum_{j\in J}\left\lfloor \sum_{i\in I}\alpha_i a_{ij}\right\rfloor x_j \leq \left\lfloor \sum_{i\in I}\alpha_i b_i\right\rfloor \tag{1.13}$$

Since Chvátal-Gomory procedure gives an exponential number of valid inequalities (1.13)*, we must add them dynamically. We thus use cut generation.*

A strong formulation may also involve an exponential number of variables. These variables usually represent partial solutions to the problem that are assembled by the constraints of the formulation. Similarly to cut generation, these variables must be incrementally added to the formulation. This procedure is named *column generation* and works as follows. First, the procedure solves the linear relaxation of the formulation, called *master problem*, restricted to a subset of variables, also known as the *restricted*

master problem. The procedure then retrieves the dual values of the constraints. Second, given the dual values, the *pricing oracle* generates the partial solution that has the best reduced cost *i.e.* that may improve the master's objective the most. Third, the algorithm adds a variable to the restricted master problem. The column is the representation of the generated partial solution in the constraints of the master. The value of the variable is equal to the number of times the partial solution is used. In order for the algorithm to converge towards the optimal solution of the master, it suffices that the pricing oracle returns, at each iteration, a negative reduced cost partial solution if one exists. The algorithm stops when the pricing oracle fails to generate a partial solution with negative (positive) reduced cost in the case of a minimization (maximization) problem.

Example 2. *(Example of column generation - Lagrangian relaxation and reformulation).*

Consider the following integer program

$$\begin{aligned} [P3] \equiv \ & \textit{minimize} && c^\intercal x && \\ & \textit{subject to} && Ax \geq a && (1.14)\\ & && Bx \geq b && (1.15)\\ & && x \in \mathbb{Z}^n_+ && \end{aligned}$$

with cost vector $c \in \mathbb{R}^n$*, right-hand sides* $a \in \mathbb{R}^{m_a}$ *and* $b \in \mathbb{R}^{m_b}$*, and coefficient matrices* $A \in \mathbb{R}^{m_a \times n}$ *and* $B \in \mathbb{R}^{m_b \times n}$*. Consider that the problem becomes much easier to solve when constraints* (1.14) *are relaxed. Let* $Z = \{x \in \mathbb{Z}^n_+ \mid Bx \leq b\}$ *be the set of feasible solutions to constraints* (1.15) *and* Q *be the index-set of solutions to* $Z = \{z^q\}_{q \in Q}$*. We perform a Lagrangian relaxation of the difficult constraints. Let* $\pi \geq 0$ *be the vector of Lagrangian multipliers associated to constraint* (1.14)*, the Lagrangian relaxation of [P3] is*

$$[LP3] \equiv \max_{\pi \geq 0} \min_{q \in Q} \left\{ c^\intercal z_q + \pi^\intercal \left(a - A z_q \right) \right\} \tag{1.16}$$

which linearization gives

$$\begin{aligned} [LP3] \equiv \max \quad & \eta && && \\ \textit{subject to} \quad & \eta \leq c^\intercal z_q + \pi^\intercal \left(a - A z_q \right) && q \in Q, && (1.17)\\ & \eta \in \mathbb{R} && && \\ & \pi \geq 0 && && \end{aligned}$$

Let λ_q be the dual variable associated to constraint (1.17). The dual problem of [LP3], called master problem, is :

$$
\begin{aligned}
[DLP3] \equiv \min \quad & \sum_{q \in Q} c^\intercal z_q \lambda_q \\
\textit{subject to} \quad & \sum_{q \in Q} A z_q \lambda_q \geq a && (1.18) \\
& \sum_{q \in Q} \lambda_q = 1 && (1.19) \\
& \lambda_q \geq 0 \quad q \in Q
\end{aligned}
$$

The latter problem involves an exponential number of variables. Since we cannot enumerate all the variables and put all of them in the formulation, we have to solve the formulation by means of column generation. We note that when variables λ can take only integer values, this formulation is equivalent to [P3].

As mentioned above, a complete enumeration of the solutions to a combinatorial problem is usually not possible. Hence, Land and Doig (1960) introduced *branch-and-bound method* to search for an optimal solution by dividing the set of all feasible solutions and pruning subsets of feasible solutions in which the optimal solution is proven to be absent using a bounding mechanism. Such a method incrementally divides the solution space by creating more and more subproblems that contain more and more integrality restrictions on variables. These restrictions are known as branching constraints. However, dividing indefinitely the solution space is equivalent to enumerate all feasible solutions. The algorithm thus maintains a primal bound and computes, for each subproblem, a dual bound by solving the linear relaxation of the subproblem. In the case of a minimization problem, the algorithm prunes a subproblem when one of the three following rules is satisfied:

- the subproblem is infeasible
- the dual bound of the subproblem is greater than the current primal bound *i.e.* there is no feasible solution better than the current one in the area described by the subproblem
- the subproblem is solved to optimality

The algorithm runs until there is no more subproblems left. The goal of these rules is to keep only the subproblems that may contain an optimal integer solution. So, the closer

the current primal bound and the dual bounds are, the fewer subproblems are generally explored. Sometimes, the solution to the linear relaxation of a subproblem is integral. If so and the value of the solution better than the current primal bound, the algorithm updates the primal bound. But when linear relaxations rarely provide integral solutions, it is worth running heuristics to search for a better feasible solution and potentially decrease the number of subproblems explored.

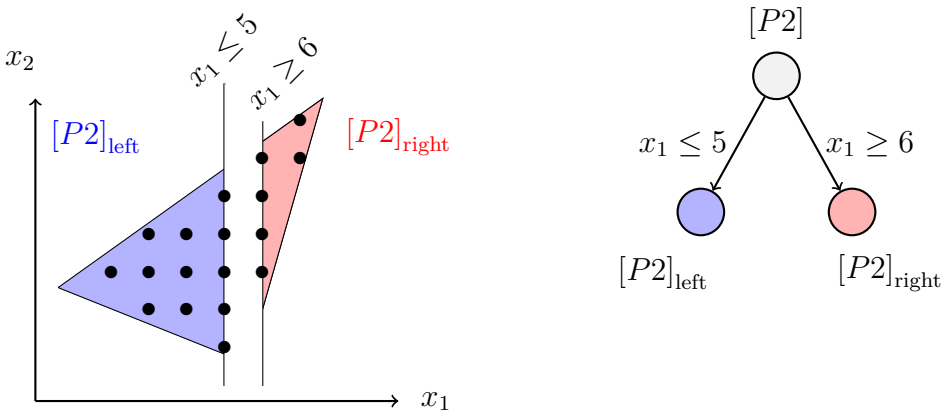


Figure 1.3: Example of a branching constraint on $[P2]$

The successive divisions of the solution space are usually illustrated by a binary tree where nodes correspond to subproblems and arcs correspond to branching constraints. The subproblem associated with a given node is the initial formulation subject to all the branching constraints met on the path from the root node to the given node. Figure 1.3 gives an example of a branch-and-bound tree for formulation $[P2]$. The value z_{LP} of the solution to the linear relaxation of [P2], computed at the root node, provides the dual bound that allows us to calculate the root gap. Given the value z^*_{IP} of the optimal integer solution to [P2], the *root gap* is equal to

$$\frac{z^*_{\text{IP}} - z_{\text{LP}}}{z_{\text{LP}}} \tag{1.20}$$

and is an indicator of the strength of the formulation.

The *branch-cut-and-price* algorithm is a branch-and-bound algorithm for an integer program that has an exponential number of variables and constraints. At each node of the branch-and-bound tree, the corresponding subproblem is thus optimized using both cut and column generation.

1.3 General structure of the branch-cut-and-price

We introduce the state-of-the-art branch-cut-and-price for the Capacitated Vehicle Routing Problem (CVRP). The structure of the algorithm mainly comes from Pecin et al. (2017a) and Sadykov et al. (2020). This algorithm will serve us as a base for the algorithms developed in this thesis. See also Costa et al. (2019) for a survey of the branch-cut-and-price algorithms applied to several types of vehicle routing problems.

A complete undirected graph $G = (\{0\} \cup \mathcal{C}, E)$ represents the transportation network. Vertex 0 is the depot, $\mathcal{C}$ denotes a set of customers. Each customer $c \in \mathcal{C}$ has a demand of d_c items. The travel costs are f_e for every edge $e \in E$. Let $\delta(V) = \{(u, v) \in E \mid u \notin V, v \in V\}$ be the set of incidents edges to a subset V of vertices. Given vertex $v \in V$, we use $\delta(v)$ as a shorthand of $\delta(\{v\})$.

A fleet of U homogeneous vehicles, each of capacity Q, delivers goods to customers. In this context, a route is an elementary cycle in G containing the depot 0 and a subset of the customers in $\mathcal{C}$. A feasible solution is a set of routes such that each customer belongs to exactly one route and the sum of the demands of the customers in each route does not exceed Q. The goal is to find a feasible solution that minimizes a total transportation cost.

We model the problem using a route-based formulation. Let R be the set of all feasible routes. Let $\tilde{x}_e^r \in \{0, 1, 2\}$ be a coefficient that is equal to the number of times edge $e \in E$ is traversed by the route $r \in R$. Let $\tilde{z}_c^r = \frac{1}{2} \sum_{e \in \delta(c)} \tilde{x}_e^r$ be equal to one if route $r \in R$ visits customer $c \in \mathcal{C}$, and 0 otherwise. Let μ_r be a binary variable equal to 1 if a vehicle is assigned to route $r \in R$, 0 otherwise. The formulation is :

$$[P4] \equiv \text{ minimize } \sum_{r \in R} \left(\sum_{e \in E} f_e \tilde{x}_e^r \right) \mu_r \tag{1.21}$$

$$\text{subject to } \sum_{r \in R} \tilde{z}_c^r \mu_r = 1 \qquad c \in \mathcal{C} \tag{1.22}$$

$$\sum_{r \in R} \mu_r \leq U \tag{1.23}$$

$$\mu_r \in \{0, 1\} \qquad r \in R \tag{1.24}$$

Objective function (1.21) minimizes the total transportation cost. Constraints (1.22) ensure that each customer is visited by exactly one route. Constraint (1.23) ensures that

the number of routes planned does not exceed the number of vehicles. Constraints (1.24) define domains of variables.

As the number of variables exponentially depends on the number of customers, the LP relaxation of [P4] is solved by a column generation approach. Route variables μ are dynamically generated by a pricing oracle. Moreover, we strengthen formulation [P4] by adding the two families of valid inequalities presented in the following section.

1.3.1 Cut generation

Rounded Capacity Cuts (RCCs) were introduced by Laporte and Nobert (1983). These cuts ensure that enough vehicles visit each subset of customers to satisfy their demands. Given a subset $C \subseteq \mathcal{C}$ of customers, term $\left\lceil \frac{\sum_{c \in C} d_c}{Q} \right\rceil$ is a lower bound on the number of vehicles required to deliver the total demand of customers in C. The cuts take the form:

$$\sum_{r \in R} \sum_{e \in \delta(C)} \tilde{x}_e^r \mu_r \geq 2 \left\lceil \frac{\sum_{c \in C} d_c}{Q} \right\rceil, \quad C \subseteq \mathcal{C}. \tag{1.25}$$

Constraints (1.25) may be separated using the CVRPSEP package (Lysgaard, 2018) which implements the heuristic by Lysgaard et al. (2004).

Rank-one cuts (R1C) are obtained by applying the Chvátal-Gomory procedure once, hence their name, on set-partitioning constraints (1.22) relaxed to set-packing constraints *i.e.* constraints $\sum_{r \in R} \tilde{z}_c^r \mu_r \leq 1, \ \forall c \in \mathcal{C}$. We recall that Chvátal-Gomory procedure is presented in Example 1. Given a vector α of multipliers such that $\alpha_c \geq 0$, $c \in \mathcal{C}$, the following rank-1 cut is valid for [P4]:

$$\sum_{r \in R} \left\lfloor \sum_{c \in \mathcal{C}} \alpha_c \tilde{z}_c^r \right\rfloor \mu_r \leq \left\lfloor \sum_{c \in \mathcal{C}} \alpha_c \right\rfloor. \tag{1.26}$$

An inequality (1.26) obtained using a vector of multipliers with l positive components is called an l-row rank-1 cut. If all positive components of α are equal, the corresponding inequality is called a subset-row cut. Jepsen et al. (2008) first introduced 3-row subset-row cuts. Pecin et al. (2017a) used l-row subset-row cuts with $l \leq 5$. General l-row rank-1 cuts with $l \leq 5$ were considered by Pecin et al. (2017b). They determined all dominant vectors of multipliers for such cuts: if an l-row rank-1 cut with $l \leq 5$ is violated, at least one rank-1 cut obtained using a dominant vector of multipliers is violated. As in Sadykov et al. (2020), the separation of l-row rank-1 cuts may be perfored by enumeration for

$l = \{1, 3\}$ and using a local search heuristic for every dominant vector of multipliers for $l = \{4, 5\}$.

1.3.2 Column generation

Let us see now how we can solve formulation [P4] by column generation. We first define the Resource Constrained Elementary Shortest Path Problem (RCESPP). We then describe a labelling algorithm that efficiently solve the RCESPP in the context of column generation. Finally, we present how we model the pricing problem as a RCESPP to generate the routes in R.

Resource constrained elementary shortest path problem

The Resource Constrained Elementary Shortest Path Problem (RCESPP) is defined over a directed graph $G'(V', A')$. Set V' contains two special vertices v_{source} and v_{sink}. Consider a set of resources denoted K. The passage of a path through an arc $a \in A'$ costs $\bar{c}_a$ and consumes κ_a^k of resource k. Paths are subject to accumulated resource consumption lower bound l_v^k and upper bound u_v^k defined on each vertex $v \in V'$ for each resource $k \in K$.

An elementary path r of length $n(r)$ is characterized by an ordered set of vertices $V(r) = (v_1^r, v_2^r \dots v_{n(r)}^r)$ in G' such that $v_1^r = v_{\text{source}}$, $v_{n(r)}^r = v_{\text{sink}}$, $v_i^r \in V'$ for $1 \leq i \leq n(r)$, where vertex is not visited more than once. The cost of path r is $\bar{c}(r) = \sum_{j=2}^{n(r)} \bar{c}_{(v_{j-1}^r, v_j^r)}$ and the accumulated consumption of resource k at vertex $v_i^r \in V(r)$ is :

$$q_k^r(v_i^r) = \begin{cases} l_{v_1^r}^k, & \text{if } i = 1 \\ \max\{l_{v_i^r}^k, q_k^r(v_{i-1}^r) + \kappa_{(v_{i-1}^r, v_i^r)}^k\}, & \text{if } i > 1 \text{ and } k \text{ disposable} \\ q_k^r(v_{i-1}^r) + \kappa_{(v_{i-1}^r, v_i^r)}^k, & \text{if } i > 1 \text{ and } k \text{ non-disposable} \end{cases} . \quad (1.27)$$

Note that the resources are *disposable* when a positive amount of a resource can be consumed to satisfy bounds on accumulated resource consumption at vertices.

The problem seeks to find the elementary path r in graph G', starting at v_{source} and ending at v_{sink}, that has the minimum cost and that respects accumulated consumption bounds at vertices *i.e.* $l_v^k \leq q_k^r(v) \leq u_v^k$ for all $v \in V(r)$ and $k \in K$.

Labelling methods, such as Dijkstra or Bellman-Ford algorithms, are a very known family of algorithms to solve shortest path problems (SPP). These methods are also useful to solve the RCESPP as pricing problem of a column generation algorithm. In the

context of a SPP, labelling algorithms build by successive adjustments a tree of partial shortest paths to find the shortest path from the source to the sink. The root of this tree is the source and each path in the tree from the source to a vertex of the graph has minimum cost. To represent the tree, these methods maintain labels. The label associated to a vertex corresponds to the shortest path from the source to the vertex (it is a partial path except if the vertex is the sink). A label contains the cost of the shortest path and the last arc used by the path. The latter entry allow us to build the path by backtracking the arcs indicated by the label at each vertex.

These methods can be adapted to the RCESPP. In this case, labels also track accumulated resource consumptions to ensure that partial paths respect resource constraints. Moreover, a label at a vertex now represents a feasible path from the source to the vertex. The labelling algorithm thus needs to store multiple labels at each vertex. In what follows, to simplify the presentation, a label L represents the (partial) path $r(L)$. A label has the following information. We denote $\bar{c}(L)$ as the cost of path r, $v(L)$ as the vertex where path r ends, $q_k(L)$ as the accumulated consumption of resource $k \in K$ at the vertex $v(L)$, and $V(L)$ as the set of vertices visited by r. In the labelling algorithm, a label has two different states : unextended or extended. If a label has the first state, the algorithm has not yet tried to extend the path described by this label. The second state is the opposite.

The most basic labelling algorithm is the enumeration of all partial paths. However, the number of labels would dramatically grow and the algorithm would become very slow. We thus use *dominance* to avoid a complete enumeration of the labels. Dominance consists in comparing the labels with each other to get rid of those that are proven not to lead to an optimum path. Consider two labels L_1 and L_2. We say that label L_2 dominates L_1 if one feasible completion of L_1 is also feasible to L_2, and the cost of the second complete path is not larger than the cost of the first. A sufficient condition to get rid of label L_1 is thus : $v(L_1) = v(L_2)$, $\bar{c}(L_2) \leq \bar{c}(L_1)$, $V(L_2) \subseteq V(L_1)$, and $q_k(L_2) \leq q_k(L_1)$ for all $k \in K$.

Algorithm 1 is a labelling algorithm. It starts at the source node and creates an initial label L_{init} such that $\bar{c}(L_{\text{init}}) = 0$, $v(L_{\text{init}}) = v_{\text{source}}$, $q_k(L_{\text{init}}) = 0\ \forall k \in K$, and $V(L_{\text{init}}) = \{v_{\text{source}}\}$. In the main iteration, the algorithm selects the label L that has the minimum resource consumptions. Then, the algorithm tries to extend the partial path corresponding to L along all arcs from the vertex $v = v(L)$ to all vertices $v' \notin V(L)$ not visited by the partial path. The concatenation of L with v' returns a new label L' such that $v(L') = v'$, $\bar{c}(L') = \bar{c}(L) + \bar{c}_{(v,v')}$, $q_k(L') = \max\{l^k_{v'}, q_k(L) + \kappa^k_{(v,v')}\}$ for $k \in K$, and

Algorithm 1 Mono-directional labelling algorithm

Let $\mathcal{U}_v \leftarrow \emptyset$ for all $v \in V'$ ▷ Sets of unextended labels
Let $\mathcal{E}_v \leftarrow \emptyset$ for all $v \in V'$ ▷ Sets of extended labels
Insert intial label L_{init} in $\mathcal{U}_{v_{\text{source}}}$
while $\bigcup_{v \in V'} \mathcal{U}_v \neq \emptyset$ **do** ▷ Main iteration
  Select label L in $\bigcup_{v \in V'} \mathcal{U}_v$
  for all $v' \notin V(L)$ **do** ▷ Loop of extensions
    $L' \leftarrow$ **concatenation**(L, v')
    if L' feasible **and** L' not dominated by any label in $\mathcal{U}_{v'} \cup \mathcal{E}_{v'}$ **then**
      Delete labels from $\mathcal{U}_{v'}$ dominated by L'
      Insert L' in $\mathcal{U}_{v'}$
    end if
  end for
  Insert L in $\mathcal{E}_v$ ▷ Mark the label as extended
  Delete L from $\mathcal{U}_v$
end while
return label of $\mathcal{E}_{v_{\text{sink}}}$ that has minimum cost

$V(L') = V(L) \cup \{v'\}$. The new label L' is feasible if and only if it satisfies accumulated resources consumptions bounds at v' *i.e.* $q_k(L') \leq u^k_{v'}$ for $k \in K$. The algorithm then checks whether the new label L' is dominated by labels associated to v'. If it is not the case, L' may be subpath of the optimum path and the algorithm gets rid of the unextended labels associated to v' that are dominated by L'. At last, the algorithm marks L as extended.

Algorithm 1 is called *mono-directional* because it starts from the v_{source} and extends the labels to build paths from the source to the sink (forward sense). We use the mono-directional algorithm to build paths from the sink to the source (backward sense) except that label extension is done differently : $q_k(L') = \min\{u_k(v'), q_k(L) - \kappa^k_{(v,v')}\}$. Given an accumulated resource consumption threshold q^* defined for a resource $k^* \in K$, a *bidirectional* labelling algorithm, proposed by Righini and Salani (2006), runs the mono-directional algorithm in forward and backward sense to get sets of forward labels $\vec{\mathcal{L}}$ and backward labels $\overleftarrow{\mathcal{L}}$ such that $q^*_k(\vec{L}) \leq q^*$ for $\vec{L} \in \vec{\mathcal{L}}$, and $q^*_k(\overleftarrow{L}) > q^*$ for $\overleftarrow{L} \in \overleftarrow{\mathcal{L}}$. Then, the algorithm tries to concatenate the forward partial paths with the backward partial paths. In contrast to the labelling algorithm, the concatenation is done on a vertex. Two labels $\vec{L}$ and $\overleftarrow{L}$ can be concatenated if $v(\vec{L}) = v(\overleftarrow{L})$, $q_k(\vec{L}) \leq q_k(\overleftarrow{L})$, and $V(\vec{L}) \cap V(\overleftarrow{L}) = \{v(\vec{L})\}$. At the end, the algorithm returns the path with smallest cost among the concatenated ones. This technique can significantly speed up the algorithm.

The graph is said *symmetric* when $\bar{c}_{(v,v')} = \bar{c}_{(v',v)}$ for $(v, v') \in V'$, all lower and upper bounds on the accumulated resource consumption are the same, and $\kappa^k_{(v,v')} = \kappa^k_{(v',v)}$ for $v, v' \in V'$ and $k \in K$. If the graph is symmetric, the backward labelling is equivalent to the forward labelling so the former can be skipped. As a result, the algorithm is up to twice faster.

Bucket-graph based labelling algorithm

Dominance checks is usually the most time consuming part of the labelling algorithm as it has quadratic complexity from the number of labels. Therefore, we use the *bucket graph based labelling algorithm*, introduced in Sadykov et al. (2020), that decreases the number of dominance checks without increasing the number of non-dominated labels.

To efficiently perform dominance checks, labels are stored in such a way that the algorithm can only test pairs of labels that potentially dominate each other. In the case of the RCESPP, labels with similar accumulated resource consumptions at the same vertex are more likely to dominate each other. Therefore, each vertex $v \in V'$ is split into buckets containing labels with similar accumulated resource consumptions. Arcs between buckets are called bucket arcs. There is a bucket arc between two buckets if a label in the tail bucket can be extended to a label in the head bucket. The resulting graph is called a *bucket graph*. For each vertex $v \in V'$, buckets are created as follows. Given a step-size Δ_k for each resource $k \in K$, the allowed range of accumulated resource k consumption at v is split into non-overlapping intervals of length Δ_k. One bucket is created for each combination of intervals from each resource. Figure 1.4 gives an example of buckets for a given vertex in a RCESPP involving two resources.

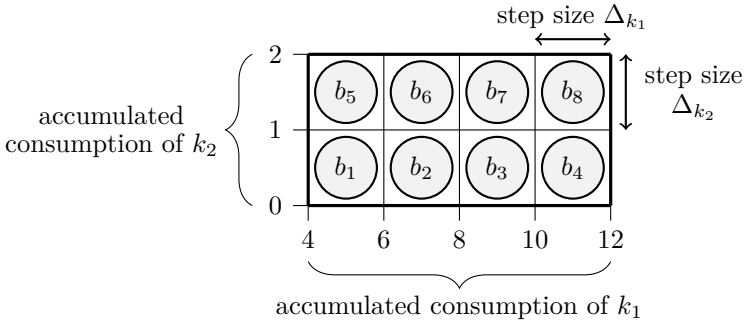


Figure 1.4: Buckets for a vertex v of a graph with two resources $K = \{k_1, k_2\}$. Bounds on accumulated resource consumption at v are $[4, 12]$ for k_1 and $[0, 2]$ for k_2. Step sizes are $\Delta_{k_1} = 2$ and $\Delta_{k_2} = 1$. At the end, this vertex gives rise to eight buckets : $b_1, b_2, b_3, b_4, b_5, b_6, b_7$ and b_8.

The algorithm performs dominance checks twice. First, when a label is created, the algorithm checks whether the new label is dominated by labels in the same bucket. Second, just before the extension of a label in bucket b, the algorithm checks whether the label is dominated by a label in other buckets. For example, given the buckets of figure 1.4, the algorithm checks whether labels in bucket b_7 are dominated by labels in buckets b_1, b_5, b_2, b_6, and b_3.

However, we keep the best costs for buckets (best cost of labels in bucket b and all buckets down and to the left). If best cost for bucket b_6 is larger than the cost of a label in b_7, a label in b_7 will not be checked for dominance by labels in b_5, b_6, b_1, b_2.

Best costs for buckets are also used to speed up concatenation.

Pricing Problem for the CVRP

As mentioned above, LP relaxation of [P4] is solved by column generation and routes variables μ are dynamically generated. We recall that a route $r \in R$ is an elementary cycle in graph G containing the depot 0 and a subset C of customers such that the total demand of C does not exceed the capacity Q of a vehicle.

Let $(\bar{\pi}, \bar{\eta}, \bar{\rho}, \bar{\xi})$ be the dual solution to the linear relaxation of $[P4]$ restricted to a subset of variables and constraints. Dual values correspond to constraints (1.22), (1.23), (1.25), and (1.26) respectively. We say that a constraint is active when it has a non-zero dual value. Let N be the set of active RCCs and let C^n be the set of customers characterizing cut $n \in N$ with dual value $\bar{\rho}_n$. Let M be the set of active R1Cs and let α^m be the vector of multipliers characterizing cut $m \in M$ that has dual value $\bar{\xi}_m$.

Now, let us see how we can model the pricing problem of [P4] as a RCESPP. To represent the distribution network, we set $V' = \{v_{\text{source}}, v_{\text{sink}}\} \cup V_{\mathcal{C}}$ where v_{source} and v_{sink} represents the depot as starting and ending point of the routes in R. Set $V_{\mathcal{C}} = \{v_c \mid c \in \mathcal{C}\}$ contains one vertex for each customer. We set $A' = (\{v_{\text{source}}\} \times V_{\mathcal{C}}) \cup \{(v, v') \mid v \neq v',\ v, v' \in V_{\mathcal{C}}\} \cup (V_{\mathcal{C}} \times \{v_{\text{sink}}\})$. Moreover, a feasible route must deliver a subset of customers that has a total demand less or equal than the capacity Q of a vehicle. We can model this constraint as a resource. We set $K = \{1\}$ because the capacity of the vehicles is the only resource of the pricing problem. We define capacity consumption on arc $a \in A' : \kappa^1_{(i,j)} = \frac{1}{2}(d_i + d_j)$ so the condition for symmetric case are satisfied. For each customer $c \in \mathcal{C}$, half of the demand is consumed on the arcs entering the vertex v_c and half of the demand is consumed on the arcs leaving the vertex v_c. Bounds on

accumulated capacity consumption are defined on vertices and limit the number of items that a vehicle can carry. Thus, for each $v \in V'$, bounds are $l_v^1 = 0$ and $u_v^1 = Q$.

Example 3. *Consider an instance with a depot* 0 *and four customers* $\mathcal{C} = \{1, 2, 3, 4\}$ *that ask for* $d_1 = 3$, $d_2 = 4$, $d_3 = 2$, *and* $d_4 = 3$ *items respectively. Vehicles can ship up to* $Q = 10$ *items. On each vertex, lower and upper bounds on accumulated capacity consumption are* 0 *and* 10 *items respectively. Figure 1.5 gives an example of a feasible route to the instance.*

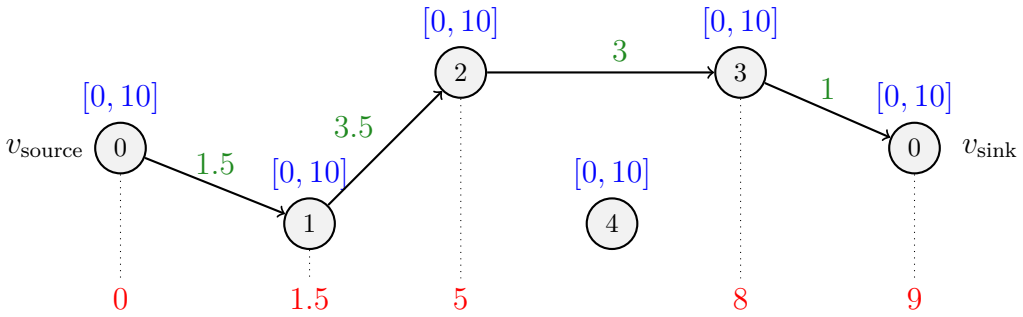


Figure 1.5: Example of a solution to the RCESPP for the CVRP. Capacity consumptions of edges are in green. Accumulated capacity consumptions at each vertex of the route are in red and bounds on accumulated capacity consumptions are in blue.

The pricing oracle must return the route $r \in R$ that has the best reduced cost. The reduced cost of route $r \in R$ is calculated as :

$$\sum_{e \in E} f_e \bar{x}_e^r - \sum_{c \in \mathcal{C}} \bar{\pi}_c \tilde{z}_c^r - \sum_{n \in N} \sum_{e \in \delta(C^n)} \frac{1}{2} \bar{\rho}_n \tilde{x}_e^r + \sum_{m \in M} \left\lfloor \sum_{c \in \mathcal{C}} \alpha_c^m \tilde{z}_c^r \right\rfloor \bar{\xi}_m \tag{1.28}$$

When there is no active R1C *i.e.* $M = \emptyset$, the reduced cost of a route can be expressed as a linear combination of reduced costs of the arcs. The reduced cost of an arc $(i, j) \in A'$ is:

$$\bar{c}_{(i,j)} = f_{(i,j)} - \frac{1}{2}(\bar{\pi}_i + \bar{\pi}_j) - \sum_{n \in N : (i,j) \in \delta(C^n)} \frac{1}{2} \bar{\rho}_n \tag{1.29}$$

We can thus directly solve the pricing problem as a RCESPP. However, the contribution of a R1C to the reduced cost of a route cannot be expressed over the arcs because of the rounding down operator. Each active R1C thus requires a new resource to allow us to compute its contribution to equation (1.28) as noticed by Jepsen et al. (2008). Given an active R1C $m \in M$ characterized by a vector α^m of multipliers, the resource k_m associated to the R1C is defined as follows. Resource consumption of both forward

and backward arcs $(v, v') \in A'$ is $\kappa^{k_m}_{(v,v')} = \alpha^m_{v'}$ if $v' \in V_{\mathcal{C}}$, and $\kappa^{k_m}_{(v,v')} = 0$ if $v' \notin V_{\mathcal{C}}$. Since this resource does not model a constraint, it does not have bounds on the accumulated resource consumption. Given a route $r \in R$, its rounded down accumulated consumption $\lfloor q^r_{k_m}(v_{\text{sink}}) \rfloor$ of resource k_m is equal to the coefficient $\lfloor \sum_{c \in \mathcal{C}} \alpha^m_c \tilde{z}^r_c \rfloor$ of the R1C $m \in M$ associated to route r.

This resource is used to compute the contribution of a R1C and not to ensure that any constraint is met. The labelling algorithm thus treats this resource differently. Consider a label L and $v = V(L)$ the vertex where it is stored. When a label L is concatenated with vertex $v' \in V'$, resource consumption is $q_{k_m}(L') = q_{k_m}(L) + \kappa^{k_m}_{(v,v')}$. However, the labelling algorithm must take into account the contribution of a R1C as soon as the accumulated resource consumption associated to the R1C is greater or equal to one. Thus, if $q_{k_m}(L') \geq 1$, then we assign following values : $\bar{c}(L') \leftarrow \bar{c}(L') + \lfloor q_{k_m}(L') \rfloor \bar{\xi}_m$ and $q_{k_m}(L') \leftarrow q_{k_m}(L') - \lfloor q_{k_m}(L') \rfloor$. Dominance remains the same except that $c(L') \leq c(L) - \sum_{m \in M : q_{k_m}(L') > q_{k_m}(L)} \bar{\xi}_m$.

All theses new resources lead to more non-dominated labels and thus make the pricing problem more difficult. To alleviate the difficulty, we use limited memory R1Cs (lm-R1Cs) proposed by Pecin et al. (2017a). A lm-R1C $m \in M$ is characterized by a vector of multipliers $(\alpha^m_c)_{c \in \mathcal{C}}$ and a subset $V^m \subseteq V'$ of vertices. The subset V^m is called memory set because when a path passes by $i \notin V^m$, the accumulated consumption of resource k_m is forgotten *i.e.* is set to 0. Example 4 illustrates the tracking of a resource associated to a lm-R1C.

Example 4. *Consider a lm-R1C $m \in M$ characterized by $\alpha^m = (0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0)$ and $V^m = \{3, 4, 5\}$. Let k_m be the resource associated to m.*

In figure 1.6, we see that the accumulated resource consumption is forgotten when the route follows arc $(5, 1)$ and leaves memory set V^m.

Moreover, we see that the reduced cost is incremented when the route reaches node 3. Indeed, the accumulated resource consumption at vertex 3 is 1. Since $q^r_{k_m}(3) \geq 1$, the reduced cost of the route is incremented by $\bar{\xi}_m$ and the value of $q^r_{k_m}(3)$ decremented by 1.

In this example, we note that the contribution of this lm-R1C is $\lfloor \frac{1}{2} \rfloor \bar{\xi}_m + \lfloor \frac{1}{2} + \frac{1}{2} \rfloor \bar{\xi}_m$ is equal to the contribution of a classic R1C which is $\lfloor \frac{1}{2} + \frac{1}{2} + \frac{1}{2} \rfloor \bar{\xi}_m$.

Since the resource of the lm-R1C is local to the memory set, it limits the increase in difficulty of the pricing problem. However, this local tracking makes a lm-R1C weaker than a R1C because a lm-R1C may have a smaller coefficient than a R1C for a given route. Consequently, given a R1C, several lm-R1Cs with different memory sets may be necessary to reach the same bound than we could get with full-memory R1C. This

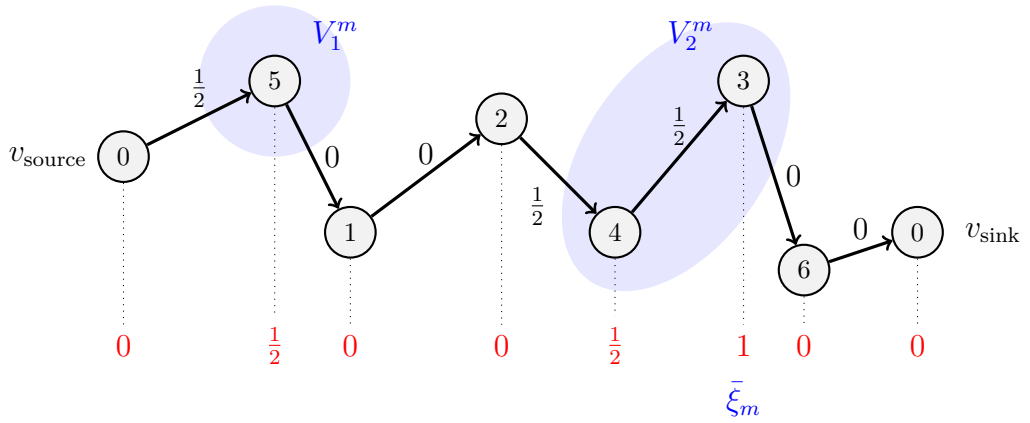


Figure 1.6: Example of the tracking on a route $r \in R$ of the resource consumption of the lm-R1C 4 with $V^m = V_1^m \cup V_2^m$. The resource consumption of edges are in black, accumulated resource consumptions at vertices are in red, and increments to the reduced cost are in blue.

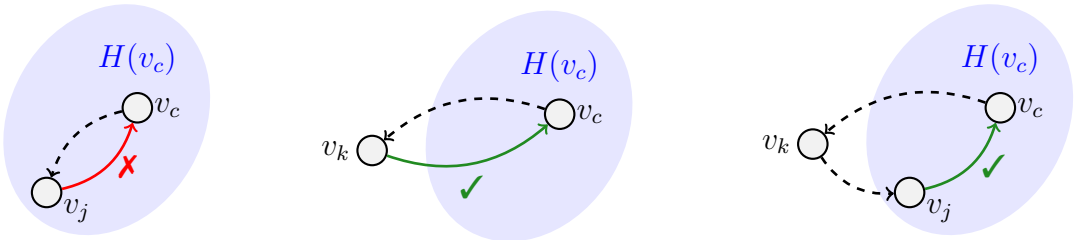
slows down the convergence of the cut-and-column generation algorithm. Although the algorithm needs more iterations to converge, the iterations are faster.

Relaxation of elementarity of paths

Since the elementarity constraint of the RCESPP is difficult and can significantly slow down the labelling algorithm, we replace it by *ng*-route relaxation introduced by Baldacci et al. (2011a). This combinatorial relaxation allows a path generated by the pricing oracle to visit a customer more than once under certain conditions. These conditions relie on the definition of a *ng*-neighbourhood for each vertex representing a customer in the graph of the pricing problem. Given a parameter $n \in \mathbb{N}$, the *ng*-neighbourhoods of $v_c \in V_{\mathcal{C}}$, denoted as $NG(v_c)$, contains v_c itself and the n customers chosen according certain criteria. In a *ng*-route, a cycle starting at customer v_c is allowed if and only if it contains a customer v_k such that $v_c \notin NG(v_k)$.

Similarly to lm-R1Cs, we can build, for each customer $c \in \mathcal{C}$, the set $H(v_c) = \{v_j \in \mathcal{C} \mid v_c \in NG(v_j)\}$ of customers having v_c in theirs *ng*-neighbourhoods. As illustrated by figure 1.7, a cycle starting at customer v_c is allowed if and only if it contains a customer v_k such that $v_k \notin H(v_c)$.

Such relaxation makes the pricing problem less difficult. Indeed, when the labelling algorithm extends a label L along v', the new label L' returned has $V(L') = (V(L) \cup \{v'\}) \cap NG(v')$ instead of $V(L') = V(L) \cup \{v'\}$. The former set is smaller which leads to a stronger dominance and so less non-dominated labels. Moreover, this relaxation is strong



(a) Cycle forbidden (b) Cycle allowed (c) Cycle allowed

Figure 1.7: Examples of cycles allowed or forbidden in a priced *ng*-route.

because when a path goes out of the memory of a node, the path is far from the node, and there is little chance that the path will return to the node. As described in Roberti and Mingozzi (2014) and Bulhoes et al. (2018), ng-neighbourhoods may be dynamically adjusted to have even stronger dual bound while keeping the pricing algorithm fast.

Reduced cost fixing

Reduced cost fixing, introduced by Dantzig, Fulkerson, and Johnson (see Grötschel and Nemhauser (2008)), consists in setting to zero the variables that are proven to be absent from any improving primal solution to the problem. In our case, reduced cost fixing is done in the pricing problem and takes place after the convergence of the column generation. This procedure may allow the pricing problem to be solved faster.

Let $R(a)$ be the set of routes in R that passes by the arc $a \in A'$. Given $a = (v, v') \in A'$, let $\bar{c}^a_{\text{best}}$ be the minimum reduced cost of the routes in $r \in R(a)$. Cost $\bar{c}^a_{\text{best}}$ is found using the bidirectional labelling algorithm, and is the sum of the best forward cost to v, the best backward cost to v', and the cost $\bar{c}_a$ of passing by arc a. Let z_{LP} be the value of the solution to the restricted master (LRP4) at a given iteration of the column generation algorithm. Let z_{IP} be the value of the best primal bound found so far. Routes in $R(a)$ are absent from any improving primal solution if and only if

$$z_{\text{LP}} + \bar{c}^a_{\text{best}} > z_{\text{IP}} \tag{1.30}$$

Arc a is removed from the pricing graph as described by Ibaraki and Nakamura (1994) and Irnich et al. (2010), or the bucket-based pricing graph as described in Sadykov et al. (2020). Since the pricing graph gets smaller, the labelling algorithm is faster. Note that the gap between z_{LP} and z_{IP} is also known as *primal-dual gap*. The smaller the primal-gap dual is, the larger the number of arcs removed is likely to be.

Enumeration of elementary routes

Since there is an exponential number of elementary routes, a complete enumeration of these routes is not possible for large instances. This is why we solve the formulation [P4] by means of column generation. However, at a given node of the branch-and-bound tree, it is sometimes possible to enumerate only the routes that may appear in an optimal solution of [P4] subject to some branching constraints. This procedure is simply called *enumeration of elementary routes* and was introduced in Baldacci et al. (2008a). This procedure allows reducing the size of the branch-and-bound tree because if the number of enumerated routes is small, we can just solve [P4] and avoid branching.

Let z_{LP} be the value of the solution to the restricted master (LRP4). Let z_{IP} be the best primal bound found so far. Let $\bar{c}_r$ be the reduced cost of route $r \in R$ calculated as in (1.28). A route r may be part of an improving solution if and only if :

$$z_{\text{LP}} + \bar{c}_r \leq z_{\text{IP}}.$$

Enumeration is performed by a special labelling algorithm different from the one used for the pricing. Here domination is performed only between labels that visited the same set of vertices. If the enumeration completes within a reasonable period of time, all the enumerated routes are added to the restricted master, and the node is terminated by MIP and then pruned. Otherwise, the enumeration stops. The smaller the primal-dual gap is, the larger is the probability that enumeration will succeed.

1.3.3 Generic BCP enhancements

Following procedures can be used regardless of the pricing problem structure.

Automatic smoothing stabilization

In the column generation algorithm, the successive resolutions of the continuous restricted master problem [LRP4] produce a sequence of dual solutions which yield dual bounds that may erratically converge towards the optimal value of [LP4]. The goal of stabilization techniques is to speed up the convergence of the column generation algorithm. According to Vanderbeck (2005), there exists three types of stabilization. First one uses a penalty function to maintain the dual solution towards a stability center, which is the best guess of an optimal dual solution. The stability center changes throughout the execution of the algorithm. Second type contains the smoothing techniques that compute

the current dual solution using previous ones. Third type of methods avoids generating extreme dual solutions.

Here, we use a stabilization technique of the second type: a smoothing stabilization with the rule introduced by Wentges (1997). The idea of this rule is to stay close to the dual solution $\hat{\pi}$ that yields the best dual bound found so far. Given an iteration t, a dual solution π^t returned by [LRP4], a parameter $\beta \in [0, 1]$, and the best dual solution $\hat{\pi}$, the dual solution $\tilde{\pi}^t$ at iteration t is calculated as :

$$\tilde{\pi}^t = \beta\hat{\pi} + (1 - \beta)\pi^t \tag{1.31}$$

The pricing problem is then called with dual solution $\tilde{\pi}^t$. Parameter β is dynamically adjusted by a heuristic devised by Pessoa et al. (2018). Basically, this heuristic increases or decreases β to get a solution $\tilde{\pi}^t$ closer to the convex combination of $\hat{\pi}$ and π^t that yields the best dual bound at the current iteration.

Multi-phase strong branching

Once the column-and-cut generation algorithm has treated a node of the branch-and-bound tree, branching constraints must be generated. To this end, we use the *multi-phase strong branching* which is a *strong branching* procedure adapted to the column generation context. This procedure was introduced by Pecin et al. (2017a). Its goal is to choose the branching constraints that improve the most the dual bound and so to reduce the size of the branch-and-bound tree.

The *strong branching* is a procedure that heuristically selects a branching constraint that potentially gives the best progress of the dual bound. The procedure first selects a collection of branching candidates based on their *pseudo-costs*. The *pseudo-cost* of a candidate heuristically estimates the progress of the dual bound in both child nodes of the branching candidate using the history of the branch-and-bound tree. Then, the procedure evaluates the progress of the dual bound in both branches of each branching candidate. This progress evaluation is stored in history. The candidate that has the largest product of dual bound improvements in the branches is chosen to be the branching constraint.

In the context of column generation, evaluating both branches of each candidate would take too much time. Therefore, only one candidate is fully evaluated after two classifying phases. This procedure is called *multi-phase strong branching*. In the first phase, the

procedure solves the continuous relaxation of the restricted master problem for both branches of each candidate. Few candidates with the largest product of dual bound improvements in the branches are chosen for the next phase. In the second phase, column generation is performed, but the pricing problem is solved with only heuristic labelling algorithms and without cut generation. The best candidate is chosen using the same product rule. In the third phase, the exact column and cut generation is performed in both branches of the chosen candidate.

Primal heuristic

As previously mentioned, having a good primal bound is important for the branch-and-bound algorithm, the reduced cost fixing procedure, and the elementary routes enumeration procedure. Therefore, at each node of the branch-and-bound tree, we use a primal heuristic that looks for primal solutions and tries to improve the primal bound.

In our work, we use the *restricted master heuristic* with a *false gap enumeration*. This procedure first enumerates elementary routes using a false primal bound z_{IP}. Value z_{IP} is artificially reduced until the enumeration succeeds. Then, enumerated columns with the smallest reduced cost are temporarily added to [P4] and a MIP solver tries to find a better feasible solution to [P4] than the best known so far.

1.3.4 Overview of the algorithm

The column and cut generation algorithm is executed at each node of the branch-and-bound tree to optimize the restricted master problem (LRP4). Figure 1.8 gives an overview of the algorithm.

First, (LRP4) is optimized using a LP solver. Second, automatic smoothing stabilization adjusts the dual solution to (LRP4). Then, the pricing oracle solves the pricing problem and returns routes that have a negative reduced cost. We use a three-stage column generation. In the first two stages, the pricing problem is solved by a heuristic version of the bucket-graph based labelling algorithm. This heuristic version consists in storing only one label in each bucket of the graph. In the last stage, the pricing problem is solved by the exact bucket-graph based labelling algorithm. If the pricing oracle does not return any routes, then column generation has converged. After the first column generation convergence, reduced cost fixing eliminates arcs in the pricing problem. Then, this procedure is performed each time the primal-dual gap decreases by more than a

given threshold. After each call to the reduced cost fixing procedure, an enumeration of elementary routes is performed for the pricing problem. If the enumeration succeeds, the pricing problem is solved by iterating over the enumerated routes (inspection) in future column generation iterations. If the total number of enumerated routes is less than a given number, all the routes are added to the formulation and the latter is solved by the MIP solver (node is terminated by MIP). Then, cut generation is performed. Cut generation is stopped either by tailing-off condition or when the time spent to solve the pricing problem is too high. Lastly, a primal heuristic seeks to improve the current primal solution and strong branching is performed.

Throughout the thesis, we will use this branch-cut-and-price algorithm as a base. For each problem considered in this thesis, we will introduce new families of cuts together with their separation algorithms. We will use different graphs and resources to model the pricing problem. We will see that the pricing problem can be decomposed into subproblems in some cases. At last, we will see that the labelling algorithm should be modified in some cases to take into account the contribution of new valid inequalities to the reduced cost of the routes.

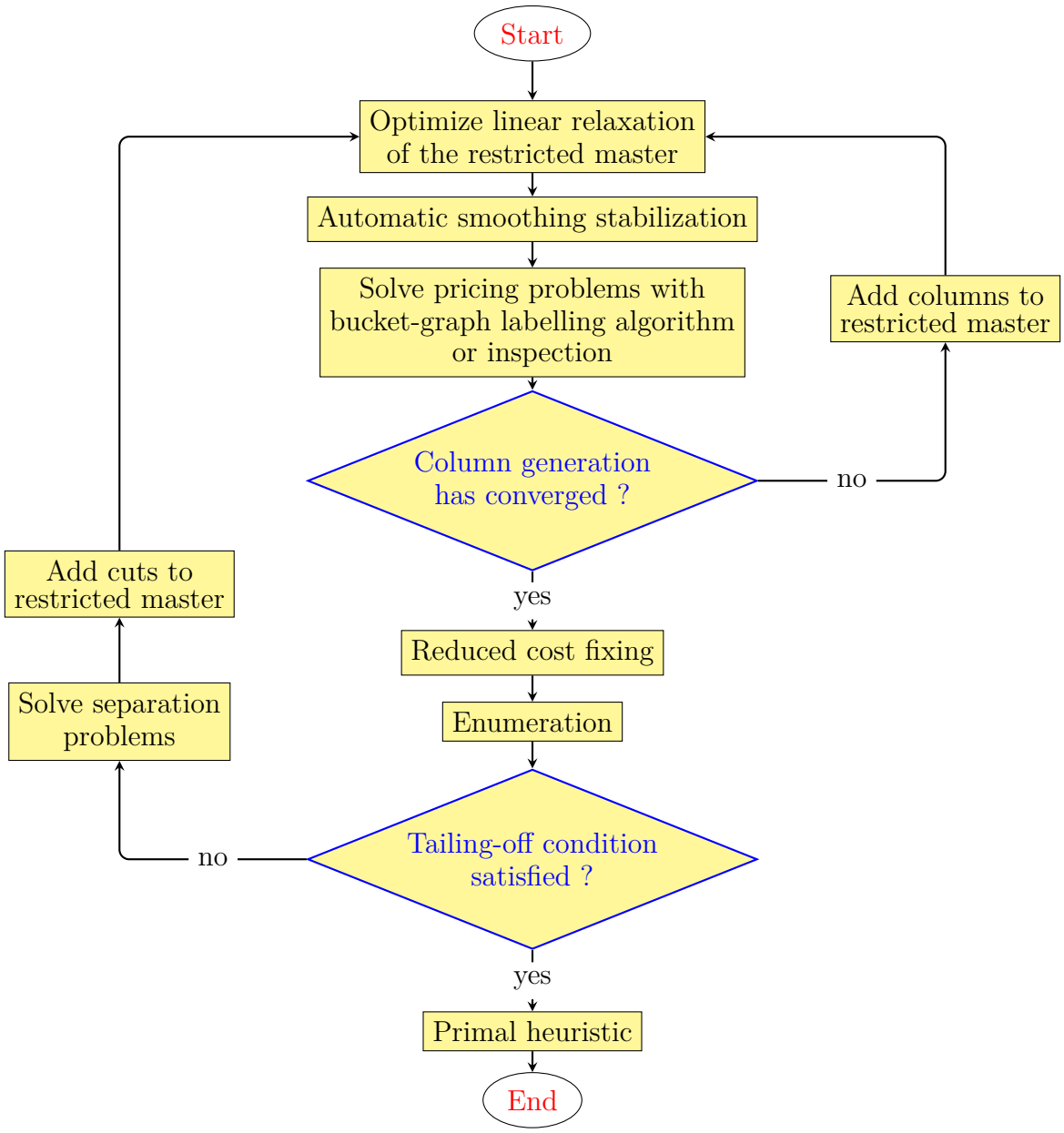


Figure 1.8: Overview of the column-and-cut generation algorithm to optimize the master problem associated to a node of the branch-and-bound tree.

Chapter 2

Two-Echelon Vehicle Routing Problem

In a context of economic globalization, the growth of urban population leads to an increase in freight transportation in cities. Freight transportation may deteriorate the quality of the urban environment with, for example, high noise levels, greenhouse gas emissions, and decrease of air quality. Moreover, freight transportation faces several constraints such as reduced access within cities and deliveries within time-windows. As a result, freight distribution patterns in city logistics change (Taniguchi and Thompson, 2002). In the past, customers were delivered straight from depots located on the outskirts of cities. Nowadays, transporters tend to use two-tier distribution systems. In the first tier, large urban trucks ships freight from warehouses or production sites to intermediate distribution facilities called satellites. In satellites, freight is processed and consolidated. Freight is then loaded in small city freighters which deliver customers located in city centers.

At the strategic level, two-tier distribution systems are considered in location-routing problems (Crainic et al., 2011). These are integrated problems in which we take decisions on both locating facilities and routing from open facilities. At tactical and operational levels, locations of depots and satellites are known. We plan only routing of vehicles. However, we should take routing decisions on both levels at the same time to devise cost-effective solutions in two-tier distribution systems. Such integration gave rise to two-echelon routing problems (Crainic et al., 2009). The first such problem proposed in the literature by Gonzalez-Feliu et al. (2007) is the two-echelon capacitated vehicle routing problem (2E-CVRP).

In the 2E-CVRP, we must determine the number of goods to be shipped from the depot to the satellites and from satellites to customers, and the optimal routes connecting

entities in each level such that vehicle capacities are not exceeded. We aim at minimizing the sum of handling costs at satellites and transportation costs depending on the total distance traveled by all vehicles.

Recently, several exact algorithms for the 2E-CVRP have been proposed in the literature. The most efficient one by Baldacci et al. (2013) is based on an enumeration of collections of first-level routes. Thus, it can efficiently tackle only instances with a small number of satellites (up to six). Moreover, this algorithm solves to optimality instances with up to 100 customers whereas the best exact algorithms for other vehicle routing problems can handle up to 300 customers (Pecin et al., 2017a). Santos et al. (2015) proposed the only branch-cut-and-price algorithm in the literature for the 2E-CVRP. This algorithm can be used to solve instances with a larger number of satellites, but experimentations show that it is less efficient than the one of Baldacci et al. (2013).

In this chapter, we propose an improved branch-cut-and-price (BCP) algorithm for the 2E-CVRP which is built on recent advances for the classical capacitated vehicle routing problem (CVRP). To further improve the efficiency of our BCP algorithm, we propose the following problem-specific enhancements:

- A new route based formulation for the problem which does not involve variables which explicitly define product flow in satellites. New level balancing constraints guarantee flow conservation in satellites.

- A new family of inequalities to improve the quality of the linear programming (LP) relaxation of the formulation. These inequalities are inspired by the depot capacity constraints introduced for the capacitated location-routing problem by Belenguer et al. (2011).

- A new branching strategy which uses variables defining the number of urban trucks visiting a subset of satellites.

To improve the current primal bound, we employed a column generation based heuristic in the course of the algorithm. Our BCP algorithm with the embedded heuristic outperformed largely the state-of-the-art exact approach by Baldacci et al. (2013) for the problem, since it solved to optimality all instances available in the literature with up to 200 customers and 10 satellites.

Finally, we generated a new set of large instances for the problem to inspire further research on the 2E-CVRP. This set involves instances with up to 300 customers and 15 satellites. These instances are derived from ones recently proposed by Schneider and Löffler (2019) for the capacitated location-routing problem.

The chapter is organized as follows. Section 2.1 reviews the literature. Section 2.2 describes the standard and the new formulations of the problem. Section 2.3 introduces the new family of satellite supply inequalities. Section 2.4 describes the proposed branch-cut-and-price algorithm. Section 2.5 reveals and discusses the computational results.

2.1 Literature review

Gonzalez-Feliu et al. (2007) first considered the 2E-CVRP. They proposed a freight-flow formulation enhanced by two families of valid inequalities. Their branch-and-cut algorithm solved to optimality instances with up to 22 customers and 2 satellites. Perboli et al. (2011) improved these results. The authors strengthened the formulation with one family of valid inequalities. They solved to optimality some instances with 33 customers. Two matheuristics were also suggested. They found feasible solutions to instances up to 50 customers with 10% of average gap from the lower bound.

Later, Jepsen et al. (2013) pointed out that the formulation in Perboli et al. (2011) is not correct for instances with more than two satellites. They proposed an alternative formulation that combines the relaxation of the split-delivery CVRP by Belenguer et al. (2000) for the first level and the model for the capacitated location routing problem by Contardo et al. (2013b) for the second level. Although this formulation is an outer approximation, its LP relaxation is stronger than the one of Perboli et al. (2011). Since this formulation has an exponential number of constraints, the authors used a branch-and-cut algorithm. A specialized branching scheme was employed to cut non-feasible integer solutions. This approach solved to optimality instances with up to 50 customers and 5 satellites. It remains the best branch-and-cut algorithm for the problem in the literature.

Contardo et al. (2012) proposed another branch-and-cut algorithm for the two-echelon capacitated location-routing problem. This problem is a generalization of the 2E-CVRP in which there are several depots and opening costs for satellites. Their branch-and-cut algorithm solved to optimality instances with up to 50 customers and 10 potential satellites.

Santos et al. (2015) proposed the first branch-cut-and-price algorithm for the 2E-CVRP. They considered a route based formulation strengthened by some valid inequalities from the CVRP literature. First-level routes are enumerated whereas second-level routes are priced by the shortest path problem with resource constraints. The pricing problem generates non-elementarity routes. They used branching strategies in the following

priority order: (1) branching on the number of vehicles traveling on a first-level route, (2) branching on the number of second-level vehicles starting from a satellite, and (3) branching on the use of an arc by a second-level route. The computational results of Santos et al. (2015) were similar to those by Jepsen et al. (2013).

The exact approach by Baldacci et al. (2013) also uses a route based formulation. The method is based on an intelligent enumeration of collections of first-level routes. Authors devised lower and upper bounding procedures to limit the number of subsets which may lead to an optimal solution. By fixing a subset of first-level routes, the problem is reduced to the multi-depot capacitated vehicle routing problem with limited depot capacities. The latter was solved by an adaptation of the algorithm by Baldacci and Mingozzi (2009). Computational experiments showed that the overall approach outperforms the one by Jepsen et al. (2013). Baldacci et al. (2013) could solve instances with up to 100 customers and 5 satellites. Their approach remains the best exact algorithm in the literature until now. However, the fact that collections of first-level routes are enumerated does not allow one to employ this approach for instances with 10 satellites or more.

There are several heuristic approaches for the 2E-CVRP in the literature. Hemmelmayr et al. (2012a) proposed an adaptive large neighbourhood search based heuristic that works for both 2E-CVRP and the location-routing problem. It largely improved the best feasible solutions found by Perboli et al. (2011). Moreover, the authors proposed a new test set of instances with up to 200 customers and 10 satellites.

Zeng et al. (2014) suggested a hybrid heuristic which is composed of a greedy randomized adaptive search procedure (GRASP) with a route-first cluster-second procedure embedded in a variable neighbourhood descent (VND). Breunig et al. (2016) developed an improved large neighbourhood-based hybrid meta-heuristic. It combines enumerative local search with destroy-and-repair principles, as well as some tailored operators to optimize the selections of satellites. Both these approaches improved the best-known solutions for many instances.

Recently, two matheuristics were proposed in the literature. Wang et al. (2017) employed a mixed-integer mathematical model for the 2E-CVRP, in which arc variables are used for the first level, and path variables for the second level. They used variable neighbourhood search to construct the set of second-level routes, and they then solved the mathematical model to improve the obtained solution. The authors improved 13 best-known solutions. Finally, Amarouche et al. (2018) used a similar approach in which a pool of routes is collected by a local search heuristic combined with a destroy-and-repair method. Then, the route based formulation is solved with the hope to improve the best

solution found so far. This approach improves 7 best-known solutions for the largest instances of the 2E-CVRP.

2.2 Formulation

Let us now formally define the problem. At the first level, a set $\mathcal{K}$ of homogeneous urban trucks ships freight from a depot denoted as 0 to a set $\mathcal{S}$ of intermediate depots, called satellites. At the second level, a set $\mathcal{L}$ of homogeneous city freighters picks freight at satellites and deliver it to a set $\mathcal{C}$ of customers. Each vehicle must return to the place from where it started its tour (depot for urban trucks and satellites for city freighters). Urban trucks have a capacity of Q_1 items, and city freighters have a capacity of Q_2 items. A satellite $s \in \mathcal{S}$ can hold up to L_s city freighters and charges f_s^H for each processed item of freight. Each customer $c \in \mathcal{C}$ asks for d_c items of freight and must be visited exactly once. At each satellite, the total amount of freight delivered by urban trucks must be equal to the amount of freight picked by city freighters that start at this satellite. The objective of the problem is to minimize the sum of handling costs f^H and transportation costs f^T.

We use the route based formulation to model the problem. The first level is similar to the split-delivery CVRP since several urban trucks can supply a satellite. However, the amount of freight delivered to each satellite is not known. A complete undirected graph $G_1 = (V_1, E_1)$, $V_1 = \{0\} \cup \mathcal{S}$, represents the first level of the distribution system. Let P be the set of feasible first-level routes and let $\tilde{z}_e^p \in \mathbb{N}$ denote the number of times path $p \in P$ uses edge $e \in E_1$.

The second level corresponds to the multi-depot CVRP where depots are satellites. Each customer is visited by one city freighter, and each satellite cannot supply more freight than the amount delivered to it by urban trucks. This level is represented by an undirected graph $G_2 = (V_2, E_2)$ where $V_2 = \mathcal{C} \cup \mathcal{S}$ and $E_2 = \{(i,j) \mid i \in \mathcal{S} \cup \mathcal{C}, j \in \mathcal{C}, i \neq j\}$. For any satellite $s \in \mathcal{S}$, let R_s be the set of feasible second-level routes starting and finishing in s. We denote $R = \cup_{s \in \mathcal{S}} R_s$. A second-level route $r \in R$ is described by vector $\tilde{x}^r$ where element $\tilde{x}_e^r \in \mathbb{N}$ denotes the number of times route r uses edge $e \in E_2$. We also introduce vector $\tilde{y}^r$ where element $\tilde{y}_c^r \in \mathbb{N}$ denotes the number of times route r visits customer $c \in \mathcal{C}$. Given graph G_i, $i = 1, 2$, we denote as $\delta_i(v)$ the set of edges in E_i incident to vertex $v \in V_i$.

The cost of traversing edge $e \in E_1 \cup E_2$ is denoted by f_e^T. From now on, we make two assumptions. First, transportation costs f^T satisfy the triangle inequality. Otherwise, we transform the instance to an equivalent one: if the minimum cost path between two

vertices v, v' passes by other vertices, we set the cost $f^T_{(v,v')}$ to the cost of the minimum path. The second assumption is that transportation costs are symmetric. If this is not the case, graphs G_1 and G_2 become directed ones, edges become arcs, and all $\tilde{z}_{e=(v,v')}$ and $\tilde{x}_{e=(v,v')}$ depending on edges are replaced by $\tilde{z}_{a=(v,v')} + \tilde{z}_{a=(v',v)}$ and $\tilde{x}_{a=(v,v')} + \tilde{x}_{a=(v',v)}$ depending on arcs. All instances in the 2E-CVRP literature satisfy these two assumptions.

2.2.1 Standard formulation

We now describe the standard route based formulation for the 2E-CVRP, used in Baldacci et al. (2013); Santos et al. (2015); Amarouche et al. (2018). Integer variable λ_p is equal to the number of urban trucks traveling on first-level route $p \in P$. We denote as S_p the set of satellites visited by route $p \in P$: $S_p = \{s \in \mathcal{S} : \sum_{e \in \delta_1(s)} \tilde{z}^p_e = 2\}$. We denote as P_S the set of first-level routes visiting at least one satellite in $S \subseteq \mathcal{S}$: $P_S = \{p \in P : S_p \cap S \neq \emptyset\}$. Continuous variable w_{ps} is equal to the amount of freight that first-level route $p \in P_{\{s\}}$ delivers to satellite $s \in S_p$. Binary variable μ_r is equal to one if and only if a city freighter travels on second-level route $r \in R$. To simplify the presentation, we introduce the continuous auxiliary variable b_s that is equal to the total amount of freight delivered to satellite $s \in \mathcal{S}$.

$$(F1) \quad \min \quad \sum_{p \in P} \sum_{e \in E_1} f^T_e \tilde{z}^p_e \lambda_p + \sum_{r \in R} \sum_{e \in E_2} f^T_e \tilde{x}^r_e \mu_r + \sum_{s \in \mathcal{S}} f^H_s b_s \tag{2.1}$$

$$\text{s.t.} \quad \sum_{r \in R} \sum_{c \in \mathcal{C}} \tilde{y}^r_c \mu_r = 1 \qquad c \in \mathcal{C} \tag{2.2}$$

$$\sum_{r \in R_s} \mu_r \leq L_s \qquad s \in \mathcal{S} \tag{2.3}$$

$$\sum_{r \in R} \mu_r \leq |\mathcal{L}| \tag{2.4}$$

$$\sum_{p \in P} \lambda_p \leq |\mathcal{K}| \tag{2.5}$$

$$\sum_{s \in S_p} w_{ps} \leq Q_1 \lambda_p \qquad p \in P \tag{2.6}$$

$$b_s = \sum_{p \in P_{\{s\}}} w_{ps} \qquad s \in \mathcal{S} \tag{2.7}$$

$$b_s = \sum_{r \in R_s} \sum_{c \in \mathcal{C}} d_c \tilde{y}^r_c \mu_r \qquad s \in \mathcal{S} \tag{2.8}$$

$$\lambda_p \in \mathbb{N} \qquad p \in P \tag{2.9}$$

$$\mu_r \in \{0, 1\} \qquad r \in R \tag{2.10}$$

$$w_{ps} \geq 0 \qquad p \in P, s \in S_p \tag{2.11}$$

Objective function (2.1) minimizes the sum of transportation and handling costs. Constraints (2.2) ensures each customer is visited by exactly one second-level route. Constraints (2.3), (2.4), and (2.5) are upper bounds on the number of used vehicles. Constraints (2.6) make sure that the capacity of urban trucks is not exceeded. Constraints (2.7) and (2.8) ensure the flow balance between the two distribution levels. Constraints (2.9), (2.10) and (2.11) define domains of variables.

2.2.2 Modified formulation

In formulation (F1), we use variable w together with constraints (2.6) and (2.7) to ensure the flow balance between two distribution levels. In the modified formulation, we replace them by another set of constraints. To simplify further the presentation, we introduce auxiliary integer variables u_S that define the number of urban trucks visiting a non-empty subset $S \subseteq \mathcal{S}$ of satellites. The modified formulation is then

$$\begin{aligned}
(F2) \quad \min \quad & (2.1) \\
\text{s.t.} \quad & (2.2) - (2.5), (2.8) - (2.10) \\
& u_S = \sum_{p \in P_S} \lambda_p \quad S \subseteq \mathcal{S}, S \neq \emptyset && (2.12) \\
& \sum_{s \in S} b_s \leq Q_1 u_S \quad S \subseteq \mathcal{S}, S \neq \emptyset && (2.13)
\end{aligned}$$

Constraints (2.12) define variables u. Level balancing constraints (2.13) replace variables w and constraints (2.6), (2.7), (2.11). Validity of constraints (2.13) follows from:

$$\sum_{s \in S} b_s \overset{(2.7)}{=} \sum_{s \in S} \sum_{p \in P_{\{s\}}} w_{ps} = \sum_{p \in P_S} \sum_{s \in S_p} w_{ps} \overset{(2.6),(2.12)}{\leq} Q_1 \sum_{p \in P_S} \lambda_p.$$

We will now prove that constraints (2.13) are sufficient to guarantee the existence of a feasible freight flow at satellites for every solution $(\bar{\lambda}, \bar{\mu})$ of formulation (F2). Remember that b and u are auxiliary variables. Their values $\bar{b}$ and $\bar{u}$ can be computed from solution $(\bar{\lambda}, \bar{\mu})$ using (2.8) and (2.12). The proof is illustrated in Figure 2.1.

Proposition 1. *For every feasible solution $(\bar{\lambda}, \bar{\mu})$ to the LP relaxation of formulation (F2) it exists a feasible solution $(\bar{\lambda}, \bar{\mu}, \bar{w})$ to the LP relaxation of formulation (F1).*

Proof. Given a solution $(\bar{\lambda}, \bar{\mu})$ and its computed values $(\bar{b}, \bar{u})$, we construct a directed graph $\bar{G} = (\bar{V}, \bar{A})$. Set $\bar{V}$ of nodes contains source $\tilde{s}$, sink $\tilde{t}$, set $\bar{P} = \{p \in P : \bar{\lambda}_p > 0\}$

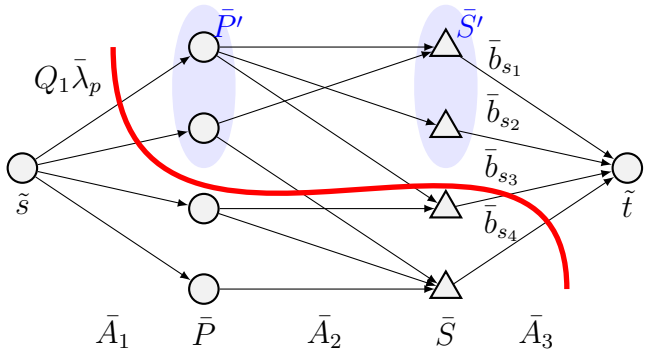


Figure 2.1: Illustration for graph $\bar{G}$ and its minimum cut in Proposition 1.

of nodes representing first-level routes in the solution, and set $\bar{S} = \{s \in \mathcal{S} : \bar{b}_s > 0\}$ of nodes representing satellites used in the solution. Set $\bar{A}$ of arcs is the union of the following three sets: $\bar{A}_1$ connects the source with $\bar{P}$, $\bar{A}_2$ connects $\bar{P}$ with $\bar{S}$, and $\bar{A}_3$ connects $\bar{S}$ with the sink. An arc $(\tilde{s}, p)$ in $\bar{A}_1$ has capacity $Q_1\bar{\lambda}_p$. An arc (p, s) belongs to $\bar{A}_2$ if and only if $s \in S_p$ and has infinite capacity. An arc $(s, \tilde{t})$ in $\bar{A}_3$ has capacity $\bar{b}_s$.

Let us now prove by contradiction that the maximum value of the $\tilde{s}$- $\tilde{t}$ flow in graph $\bar{G}$ is equal to $\sum_{c\in\mathcal{C}} d_c = d(\mathcal{C})$. Suppose that the maximum flow is strictly less than $d(\mathcal{C})$. Let $\bar{V}'$ be the subset of $\bar{V}$ obtained from a minimum $\tilde{s}$-$\tilde{t}$ cut in $\bar{G}$, $\tilde{s} \in \bar{V}'$. Let $\bar{S}' = \bar{V}' \setminus \bar{S}$ and $\bar{P}' = \bar{V}' \setminus \bar{P}$. We denote as $\delta(\bar{V}') = \{(v, v') \in \bar{A} \mid v \in \bar{V}', v' \notin \bar{V}'\}$ the set of arcs forming the minimum cut. From the supposition and the max-flow-min-cut theorem it follows that the total capacity of $\delta(\bar{V}')$ is less than $d(\mathcal{C})$. Thus $\delta(\bar{V}')$ contains at least one arc in $\bar{A}_1$ and does not contain all arcs in $\bar{A}_3$. Therefore, the total capacity of arcs in $\bar{A}_1 \cap \delta(\bar{V}')$ is strictly less than the total capacity of arcs in $\bar{A}_3 \setminus \delta(\bar{V}')$:

$$\sum_{p\in\bar{P}'} Q_1\bar{\lambda}_p < \sum_{s\in\bar{S}'} \bar{b}_s. \tag{2.14}$$

Set $\delta(\bar{V}')$ does not contain any arc in $\bar{A}_2$ as they have infinite capacity. Thus $\bar{V}' \cap \bar{P} \cap P_{\bar{S}'}$ is empty, and $\bar{\lambda}_p = 0$ for all $p \in P_{\bar{S}'} \setminus \bar{P}'$. From the latter and (2.14), it follows that $\sum_{s\in\bar{S}'} \bar{b}_s > Q_1 \sum_{p\in P_{\bar{S}'}} \bar{\lambda}_p$ which violates constraints (2.12) and (2.13) for set $\bar{S}'$ of satellites. Thus $(\bar{\lambda}, \bar{\mu})$ is not a feasible solution to the LP relaxation of (F2) which leads to a contradiction.

We have just proved that the maximum flow in graph $\bar{G}$ has value $d(\mathcal{C})$. We now set $\bar{w}_{ps}$ equal to the flow from $p \in \bar{P}$ to $s \in \bar{S}$ for every $(p, s) \in \bar{A}_2$, and to 0 otherwise. By construction of graph $\bar{G}$, constraints (2.6) and (2.7) are satisfied by $(\bar{\lambda}, \bar{b}, \bar{w})$, and $(\bar{\lambda}, \bar{\mu}, \bar{w})$ is a feasible solution to the LP relaxation of (F1). □

Since the number of constraints (2.12) and (2.13) is exponential, we should generate them dynamically. The proof of Proposition 1 gives a method to separate both integer and fractional solutions of (F2). In the separation procedure, we search for a minimum cut in graph $\bar{G}$ constructed from $\bar{\lambda}$ and $\bar{\mu}$. Once set $\bar{S}'$ of satellites is found, it is further separated into subsets such that there is no path in $\bar{P}$ visiting two satellites in different subsets. Then, we add constraints (2.12) and (2.13) for every such subset of satellites.

2.2.3 Valid inequalities

In our BCP algorithm, we use four families of valid inequalities. The first two families were introduced in Chapter 1 : RCCs et lm-R1Cs. In this section, we present the third family and some valid lower bounds on the number of vehicles used. To simplify the presentation, we introduce auxiliary variables x and y:

$$x_e^s = \sum_{r \in R_s} \tilde{x}_e^r \mu_r, \quad s \in \mathcal{S},\ e \in E_2, \qquad y_c^s = \sum_{r \in R_s} \tilde{y}_c^r \mu_r, \quad s \in \mathcal{S},\ c \in \mathcal{C}.$$

Integer variable x_e^s is equal to the number of times edge $e \in E_2$ is used by city freighters started from satellite $s \in \mathcal{S}$. Binary variable y_c^s is equal to one if and only if customer $c \in \mathcal{C}$ is visited by a city freighter started from satellite $s \in \mathcal{S}$.

Visited satellite inequalities

A customer can be visited by a route $r \in R_s$ only if satellite s is visited by at least one urban truck. Therefore, next visited satellite inequalities (VCI) are valid:

$$y_c^s \leq u_{\{s\}}, \quad c \in \mathcal{C}, s \in \mathcal{S}. \tag{2.15}$$

Although inequalities (2.15) are rather straightforward, we did not find any work in the literature which uses them. Separation of constraints (2.15) is performed by enumeration of all pairs $(c, s) \in \mathcal{C} \times \mathcal{S}$.

Lower bounds on the number of vehicles

We define lower bounds on the number of urban trucks

$$\sum_{p \in P} \lambda_p \geq \left\lceil \frac{d(\mathcal{C})}{Q_1} \right\rceil \tag{2.16}$$

and the number of city freighters

$$\sum_{r \in R} \mu_r \geq \left\lceil \frac{d(\mathcal{C})}{Q_2} \right\rceil \tag{2.17}$$

Moreover, a subset S of satellites must be visited by enough urban trucks to supply the demand that cannot be delivered from satellites in $\mathcal{S} \setminus S$. Therefore, the next lower bound on u_S is valid :

$$u_S \geq \left\lceil \frac{d(\mathcal{C}) - \sum_{s \in \mathcal{S} \setminus S} L_s Q_2}{Q_1} \right\rceil, \quad S \subset \mathcal{S} \tag{2.18}$$

Inequalities (2.18) are useful when the number of city freighters that can start from satellites is limited ($L_s < |\mathcal{L}|, s \in \mathcal{S}$).

2.3 New family of valid inequalities

We propose a new family of satellite supply inequalities (SSI) inspired by the depot capacity constraints introduced for the capacitated location-routing problem by Belenguer et al. (2011). To simplify the presentation below, we denote $C^{\complement} = \mathcal{C} \setminus C$ and $S^{\complement} = \mathcal{S} \setminus S$. Let us introduce SSI through an example.

Example 5. *Consider urban trucks with capacity $Q_1 = 10$ and city freighters with capacity $Q_2 = 6$. Figure 2.2 shows a fractional solution $(\bar{u}, \bar{\mu})$ to the LP relaxation of (F2). Here $\mathcal{S} = \{s_1, s_2\}$ and set $\mathcal{C}$ contains seven customers. Consider subset C of customers with $d(C) = 11$. Consider also subset $S = \{s_2\}$ of satellites with $\bar{u}_S = 1$. Clearly, S can supply only demand of at most $Q_1 \bar{u}_S = 10$ units and thus cannot supply set C of customers alone. In this fractional solution, C is supplied by 1.8 city freighters coming from S and 0.2 city freighters coming from $S^{\complement}$. The violated SSI states that either two or more urban trucks should visit $S = \{s_2\}$ or at least one city freighter coming from satellites in $S^{\complement} = \{s_1\}$ should visit some customers in C:*

$$\sum_{s \in S^{\complement}} \sum_{e \in \delta(C)} x_e^s \geq 2 \cdot (2 - u_S) .$$

2.3.1 Satellite supply inequalities

We now define $g^C(u)$ as the function which gives a lower bound on the number of city freighters required to cover the demand of a subset C of customers that $\lfloor u \rfloor$ urban trucks

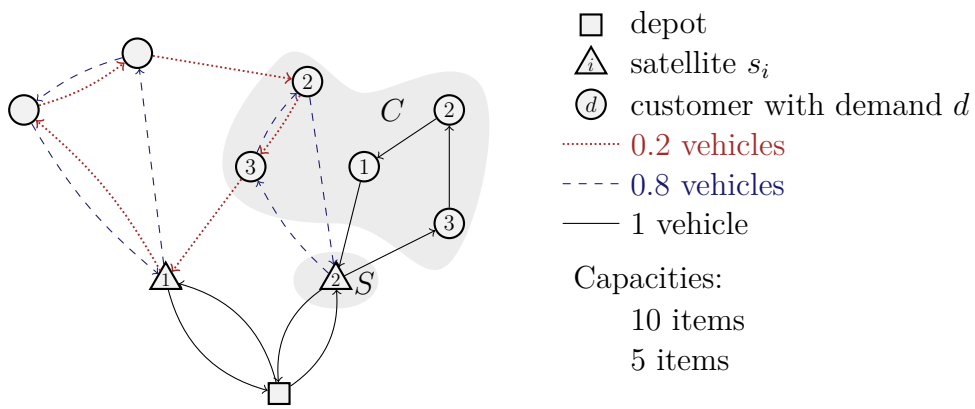


Figure 2.2: Example of a satellite supply inequality, violated for given S and C.

cannot supply:

$$g^C(u) = \max\left\{0, \left\lceil \frac{d(C) - Q_1\lfloor u \rfloor}{Q_2} \right\rceil\right\}.$$

Proposition 2. *Given $C \subset \mathcal{C}$ and $S \subset \mathcal{S}$, the following inequality is valid for the 2E-CVRP*

$$\sum_{s \in S^{\complement}} \sum_{e \in \delta_2(C)} x_e^s \geq 2 \cdot g^C(u_S). \tag{2.19}$$

Proof. Consider a feasible solution $(\bar{x}, \bar{b}, \bar{u})$ of formulation (F2). The following rounded capacity inequality is satisfied by $\bar{x}$ and $\bar{b}$:

$$\sum_{s \in S^{\complement}} \sum_{e \in \delta_2(C)} \bar{x}_e^s \geq 2 \left\lceil \frac{d(C) - \sum_{s \in S} \bar{b}_s}{Q_2} \right\rceil \tag{2.20}$$

From constraint (2.13) and the integrality of variable $\bar{u}_S$, it follows

$$\sum_{s \in S} \bar{b}_s \leq Q_1 \bar{u}_S = Q_1 \lfloor \bar{u}_S \rfloor. \tag{2.21}$$

By combining (2.20) and (2.21) we obtain that (2.19) is satisfied by $\bar{x}$ and $\bar{u}$. □

Function $g^C(u)$ is not linear and cannot be used directly. Instead, we use the piecewise linear function, denoted as $h^C(u)$, which forms the convex hull of the epigraph of $g^C(u)$. We denote as $\tilde{u}^C$ the ordered vector of (integer) values u of extreme points of h^C:

$$\tilde{u}^C = \left(\tilde{u}_0^C = 0, \tilde{u}_1^C, \ldots, \tilde{u}_{k(C)}^C = \lceil d(C)/Q_1 \rceil\right).$$

Figure 2.3 depicts an example of functions g^C and h^C for $Q_1 = 10$, $Q_2 = 4$, and $d(C) = 32$. In the left plot, the epigraph of g^C is the grey area. In the right plot, function

h^C is the bold line. Extreme points of h^C are H_0, H_2, H_3, H_4, but not H_1. Therefore, $\tilde{u}^C = (0, 2, 3, 4)$, and $k(C) = 3$.

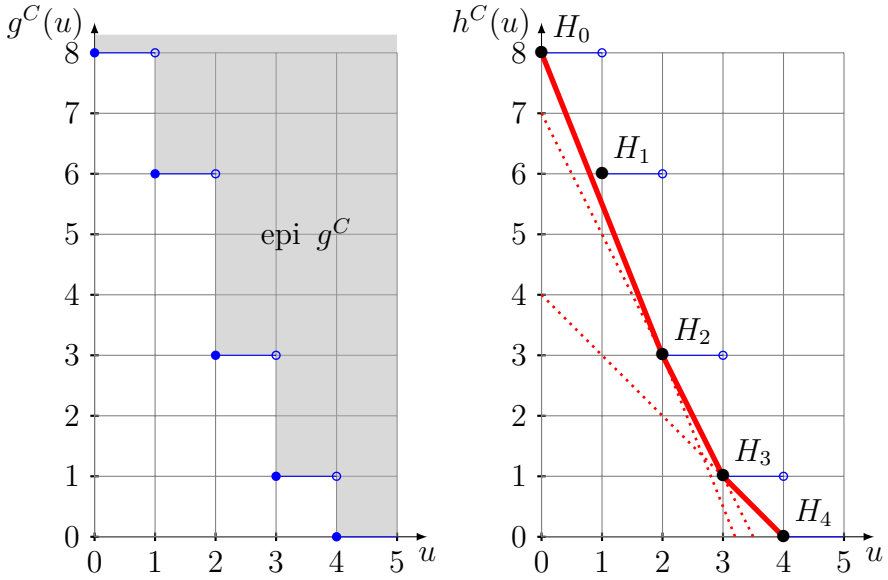


Figure 2.3: Example of functions g^C (on the left) and h^C (on the right).

Proposition 3. *Given subsets $C \subset \mathcal{C}$, $S \subset \mathcal{S}$, and an integer $0 < k \leq k(C)$, the following inequality is valid for the 2E-CVRP*

$$\sum_{s \in S^{\complement}} \sum_{e \in \delta_2(C)} x_e^s \geq 2 \cdot \left(h^C(\tilde{u}_{k-1}^C) - \frac{h^C(\tilde{u}_{k-1}^C) - h^C(\tilde{u}_k^C)}{\tilde{u}_k^C - \tilde{u}_{k-1}^C} (u_S - \tilde{u}_{k-1}^C) \right) \tag{2.22}$$

Proof. The right-hand side of each constraint (2.22) corresponds to a linear piece of function h^C. Thus the proof follows from Proposition 2 and from the fact that $h^C(u) \leq g^C(u)$ for all $u \geq 0$. □

2.3.2 Separation of Satellite supply inequalities

Let $(\bar{x}, \bar{u})$ be the values of variables x and u in a solution to the LP relaxation of (F2). The following problem finds the most violated inequality.

$$\max_{S \subset \mathcal{S}, C \subset \mathcal{C}} 2 \cdot h^C(\bar{u}_S) - \sum_{s \in S^{\complement}} \sum_{e \in \delta_2(C)} \bar{x}_e^s \tag{2.23}$$

The first and second terms of the objective function are non-linear functions of C and S. Thus enumeration of C and S is required to compute (2.23) exactly using an integer program. Since the number of subsets is exponential, we propose a heuristic to

separate SSI. Although the heuristic does not necessarily find the most violated inequality, it offers a good trade-off between the computational effort and the violation of found inequalities. Our heuristic works with a fixed set of satellites. The following proposition gives a dominance rule which will allow us to discard non-interesting subsets of satellites.

Proposition 4. *Consider a solution $(\bar{x}, \bar{u})$ to the LP relaxation of (F2) and a fixed set C of customers. If SSI (2.22) is violated for C and a set S_1 of satellites, then it is violated for C and any set $S_2 \supseteq S_1$ such that $\bar{u}_{S_2} = \bar{u}_{S_1}$.*

Proof. The right-hand side of (2.22) is fixed for a fixed set C of customers and a fixed value u_S. Thus right-hand side of the SSIs for pairs (C, S_1) and (C, S_2) is the same. For fixed values of variables x, the left-hand side of the SSI for pair (C, S_2) is not larger than one of the SSI for pair (C, S_1), as $S_2^{\complement} \subseteq S_1^{\complement}$. Thus the violation of the SSI for pair (C, S_2) is not smaller than one of the SSI for pair (C, S_1). □

To enumerate all the non-dominated sets of satellites, we first build the power set $\bar{\mathcal{U}}$ of set $\bar{S}$ of satellites used in the solution. Since we look for the largest subsets of satellites, we append all satellites in $\mathcal{S} \setminus \bar{S}$ to each set in $\bar{\mathcal{U}}$. Finally, we exclude from $\bar{\mathcal{U}}$ all sets S_1 such that there exists $S_2 \in \bar{\mathcal{U}}$ with $S_2 \supseteq S_1$ and $\bar{u}_{S_2} = \bar{u}_{S_1}$. This is done by the exhaustive enumeration as the cardinality of set $\bar{\mathcal{U}}$ is not large for the instances of the literature.

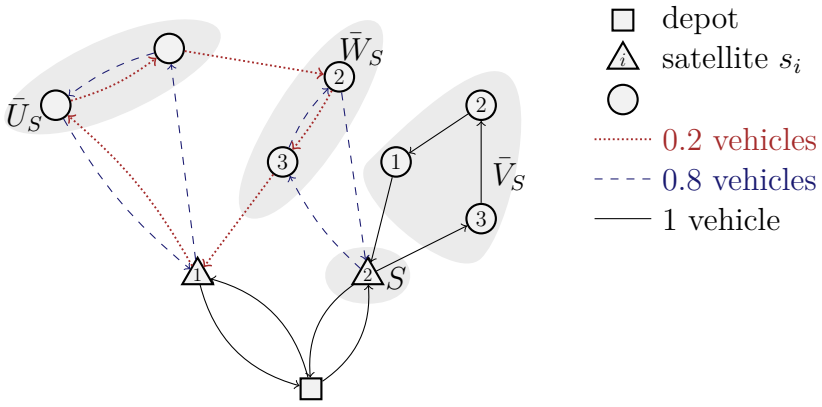


Figure 2.4: Separation graph $\bar{G}_2(S)$ for the fractional solution in Example 5

Given a set $S \in \bar{\mathcal{U}}$ of satellites, we now look for subsets of customers that violate SSI. We split customers in three subsets. Let $\bar{U}_S \subset \mathcal{C}$ be the set of customers visited only by routes started from $S^{\complement}$, let $\bar{V}_S \subset \mathcal{C}$ be the set of customers visited only by routes started from S, and let $\bar{W}_S = \mathcal{C} \setminus (\bar{U}_S \cup \bar{V}_S)$. Figure 2.4 illustrates the partition of customers of Example 5. We then build graph $\bar{G}_2(S)$ in the following way.

- Graph $\bar{G}_2(S)$ is the subgraph of G_2 induced by vertices in $\bar{U}_S \cup \bar{W}_S \cup S^{\complement}$.

- Weight of each edge e in $\bar{G}_2(S)$ is equal to $\sum_{s \in S^{\complement}} \bar{x}_e^s$.

- Set $\bar{U}_S \cup S^{\complement}$ of vertices in $\bar{G}_2(S)$ is merged into one vertex $\bar{s}$ by successively contracting all edges having two incident vertices in $\bar{U}_S \cup S^{\complement}$. Weight of each edge $(\bar{s}, c)$, $c \in \bar{W}_S$, in final graph $\bar{G}_2(S)$ is then equal to $\sum_{s \in S^{\complement}} \sum_{i \in \bar{U}_S \cup S^{\complement}} \bar{x}_{(i,c)}^s$.

Afterward, we compute the minimum cut in $\bar{G}_2(S)$. Let $\bar{C}_S$ be the subset of vertices obtained from the minimum cut, $\bar{s} \notin \bar{C}_S$. First, we verify whether SSI based on set S of satellites and set $C = \bar{C}_S \cup \bar{V}_S$ of customers is violated. Afterward, we iteratively enlarge set C in a greedy manner and check the violation of SSI based on S and C at each iteration. Customer c' to include in current set C (and exclude from $\bar{W}_S$) in each iteration is

$$c' = \arg\max_{c \in \bar{W}_S} \left\{ \frac{\sum_{s \in S^{\complement}} d_c \bar{y}_c^s}{\sum_{s \in S^{\complement}} \sum_{i \in \bar{U}(S,C,c)} \bar{x}_{(c,i)}^s} \right\}, \tag{2.24}$$

where $\bar{U}(S, C, c) = S^{\complement} \cup \bar{U}_S \cup \bar{W}_S \setminus \{c\}$. The intuition behind (2.24) is that we try to increase the first term of (2.23) while increasing not too much the second term. The separation procedure is formally given in Algorithm 2.

Algorithm 2 Separation procedure for the satellites supply inequalities

```
We are given (λ̄, μ̄) and computed entities P̄, ū, x̄, ȳ
I is the set of found violated SSI
Find set Ū of non-dominated subsets of satellites
for all S ∈ Ū do
    Find Ū_S, V̄_S, W̄_S and build graph Ḡ_2(S)
    Compute the minimum cut in Ḡ_2(S) and corresponding set C̄_S
    C ← C̄_S ∪ V̄_S
    repeat
        If SSI based on S and C is violated by (ū, x̄), add the inequality to I
        Find customer c' in W̄_S using (2.24)
        C ← C ∪ {c'}
        W̄_S ← W̄_S \ {c'}
    until W̄_S = ∅
end for
Return a specified number of the most violated SSI in I
```

2.4 Branch-cut-and-price algorithm

Formulation (F2) together with valid inequalities (1.25), (1.26), (2.15), and (2.22) is solved by an adaptation of the branch-cut-and-price algorithm presented in Section 1.3.4. In this section, we describe how we model the pricing problem, the parameters of the BCP, and the branching rules.

As the number of variables depends exponentially on the number of satellites and customers, the LP relaxation of (F2), which we call the *master problem*, is solved by the column and cut generation approach. As we consider instances with at most 15 satellites, all first-level route variables λ are added to the formulation from the start of the algorithm. All exact algorithms in the literature which use the route based formulation for the 2E-VRP follow the same approach.

2.4.1 Pricing problem

Second-level route variables μ are dynamically generated by solving the pricing problem. It is decomposed into $|\mathcal{S}|$ subproblems (SP_s), one per satellite $s \in \mathcal{S}$. The set of feasible solutions to the pricing subproblem (SP_s) is the set R_s of paths in graph G_2. A path r belongs to R_s if and only if :

- it starts and finishes in vertex s: $\sum_{e \in \delta_2(s)} \tilde{x}_e^r = 2$;
- it does not pass through other satellites: $\sum_{e \in \delta_2(\mathcal{S} \setminus \{s\})} \tilde{x}_e^r = 0$;
- it passes through each customer at most once: $\tilde{y}_c^r \leq 1$, $\forall c \in \mathcal{C}$.
- its total delivered demand does not exceed the capacity of a city freighter: $\sum_{c \in \mathcal{C}} \tilde{y}_c^r \leq Q_2$.

Let π, ψ, ϕ, ρ, τ, θ, ξ, and ζ be optimum dual values for the master problem restricted to a subset of variables μ. These dual values correspond to constraints (2.2), (2.3), (2.4), (2.8), (1.25), (1.26), (2.15), and (2.22). Let also K be the collection of active rank-1 cuts corresponding to vectors $(\alpha^k)_{k \in K}$, and M_s be the collection of active SSI based on sets $(S_m, C_m)_{m \in M_s}$, $s \in S_m^{\mathcal{C}}$. The reduced cost of a path $r \in R_s$ is then equal to

$$\begin{aligned} &\sum_{e \in E_2} \left(f_e^T - \sum_{\substack{C \in \mathcal{C}: \\ e \in \delta(C)}} \tau_C - \sum_{\substack{m \in M_s: \\ e \in \delta_2(C_m)}} \zeta_m \right) \tilde{x}_e^r \\ &- \sum_{c \in \mathcal{C}} \sum_{e \in \delta_2(c)} \frac{1}{2} \left(\pi_c + d_c \rho_s - \xi_c^s \right) \tilde{x}_e^r + \sum_{k \in K} \theta_k \left\lfloor \sum_{c \in \mathcal{C}} \alpha_c^k \tilde{y}_c^r \right\rfloor + \psi_s + \phi. \end{aligned} \tag{2.25}$$

Each pricing subproblem (SP_s) is solved by the bucket graph based bi-directional labelling algorithm proposed by Sadykov et al. (2020) and presented in chapter 1.

2.4.2 Column and cut generation

We recall that we use three-stage column generation. For each subproblem, the heuristic pricing at the first two stages generates at most 30 columns, and exact pricing at the third stage generates at most 150 columns.

The reduced cost fixing procedure, presented in Section 1.3.2, is first performed after the first column generation convergence, and then each time the primal-dual gap decreases by more than 10%. Note that graph reduction is not the same in different pricing subproblems.

If the total number of routes enumerated in all subproblems by the enumeration procedure presented in Section 1.3.2 is less than 5000, all the routes are added to formulation (F2) and the latter is solved by the MIP solver.

We define variables u only for subsets with 5 satellites or less in order to limit the size of the formulation and the number of candidates for branching. Moreover, we define variables $u_S = t_1 - \sum_{p \notin P_S} \lambda_p$ where t_1 is the total number of urban trucks used. It allows us to keep the coefficient matrix sparse. Variables u_S are replaced by $t_1 - \sum_{p \notin P_S} \lambda_p$ when $|S| \geq 6$. In the beginning, all constraints (2.13) are added to formulation (F2) for instances with at most 10 satellites. For other instances, only constraints (2.13) for sets $S \subset \mathcal{S}$, $|S| \leq 5$, are added. Other constraints are dynamically separated as described in Section 2.2.2.

In each cut generation round, we add at most 100 rounded capacity cuts (1.25), 450 rank-1 cuts (1.26), 50 VCI (2.15), and 150 SSI (2.22) to the master problem. The cut generation is stopped either by the tailing-off condition or when the time spent to solve at least one pricing subproblem exceeds 1 second. The tailing-off condition is satisfied when after 3 cut generation rounds the primal-dual gap decreases by less than 2% per round.

2.4.3 Branching

We perform branching on: the number of first-level routes visiting a subset of satellites (variables u), the use of first-level routes (variables λ), the use of an edge in E_2 (variables x), the assignment of a customer to a satellite (variable y), the number of first-level routes ($\sum_{p \in P} \lambda_p$), the number of second-level routes ($\sum_{r \in R} \mu_r$), and the number of second-level routes started from a satellite s ($\sum_{r \in R_s} \mu_r$). We use a multi-phase strong branching

procedure, similar to Sadykov et al. (2020), to choose the most promising branching candidate.

We use the multi-phase strong branching procedure presented in Section 1.3.3. The branching procedure first chooses at most 50 branching candidates and up to half of the candidates are chosen according to the branching history using pseudo-costs. Three candidates are chosen for the second phase.

2.4.4 Primal heuristic

After each node in the branch-and-bound tree, a heuristic looks for improving feasible solutions to the 2E-CVRP. We first tried the standard restricted master heuristic Sadykov et al. (2018) in which the MIP solver solves the current restricted master problem. However, we have not been satisfied with the performance of this heuristic, especially for instances with large capacity of city freighters. The reason is that sometimes only a small part of columns in the restricted master are elementary, thus making the solution space of the restricted master very small or even empty.

Instead, we use the heuristic based on an artificial primal bound and the elementary route enumeration presented in Section 1.3.3. This heuristic returns 10000 elementary routes with the smallest reduced cost and we add them to the master problem. Finally, IBM CPLEX MIP solver tries to solve for the resulting problem within $|\mathcal{C}|/2.5$ seconds. We activate the polishing heuristic Rothberg (2007) implemented in CPLEX.

2.5 Computational results

The model and the separation algorithms for constraints (2.12), (2.13), VCI (2.15), and SSI (2.22) were implemented in Julia 0.6 language using JuMP Dunning et al. (2017) and LightGraphs packages. We also used:

- BaPCod C++ library Vanderbeck et al. (2019) which implements the BCP framework;
- C++ code, developed by Sadykov et al. (2020), which implements the bucket graph based labelling algorithm, bucket arc elimination procedure, elementary route enumeration, and the separation of limited-memory rank-1 cuts;
- CVRPSEP C++ library Lysgaard (2018) which implements heuristic separation of rounded capacity cuts;

- IBM CPLEX Optimizer version 12.8.0 as the LP solver in column generation and as the solver for the enumerated MIPs.

Experiments were run on a 2 Dodeca-core Haswell Intel Xeon E5-2680 v3 servers at 2.5 GHz. On each server, we solved 24 instances with up to 200 customers and up to 10 satellites that share 128Go of RAM. Larger instances having either 300 customers or 15 satellites were solved by batches of 4 instances sharing 128Go of RAM. Each instance is solved on a single thread.

2.5.1 Instances

Table 2.1 shows the sets of instances from the literature that we used. Constraints (2.3) limiting the number of city freighters per satellite are required only for set 4B. Therefore, constraints (2.18) are useful only for set 4B. Set 5 duplicates each instance: the first instance has the standard capacity of city freighters, and the second one, with suffix "b", has the double capacity. Only set 6B has non-zero handling costs. We do not consider instances of set 3, proposed in Gonzalez-Feliu et al. (2007), as they are easily solved by our algorithm and by Baldacci et al. (2013).

Table 2.1: Sets of 2E-CVRP instances from the literature used for experiments

Set	#	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{C}\rvert$	Notes	Authors
4A	54	2, 3, 5	50	$L_s < \lvert\mathcal{L}\rvert$	Crainic et al. (2010)
4B	54	2, 3, 5	50		Crainic et al. (2010)
5	18	5, 10	100, 200	low and high Q_2	Hemmelmayr et al. (2012a)
6A	27	4, 5, 6	50, 75, 100		Baldacci et al. (2013)
6B	27	4, 5, 6	50, 75, 100	$f_s^H > 0$	Baldacci et al. (2013)

As all instances were solved to optimality by our BCP algorithm, we generated 51 additional instances involving up to 300 customers and 15 satellites. They are based on instances of families a, b, and c proposed by Schneider and Löffler (2019) for the capacitated location-routing problem. In comparison with the original instances, we added the position of the depot, capacity of urban trucks, the number of urban trucks, and the number of city freighters. We put the depot at location $(0, 0)$. We set $Q_1 = 9 \cdot Q_2$, $|\mathcal{K}| = \lceil 1.75 \cdot d(\mathcal{C})/Q_1 \rceil$, and $|\mathcal{L}| = \lceil 2.5 \cdot d(\mathcal{C})/Q_2 \rceil$.

2.5.2 Experimental analysis of BCP variants

In the first experiment, we compare different variants of our BCP algorithm for solving the 54 largest literature instances from sets 5, 6A, and 6B with 75, 100, and 200 customers.

To eliminate randomness related to improvement of primal bounds, all variants were executed without primal heuristic and with the initial primal bound equal to the optimum solution value (plus small ϵ) for each instance. We tested the following variants.

BCP$_0$ — the base variant which uses formulation (F1), without separating VCI and SSI, and without branching on variables u. It can be considered as a straightforward adaptation of the BCP algorithm in Sadykov et al. (2020) for solving the 2E-CVRP.

BCP$_{+u}$ — the base variant with branching on variables u.

BCP$_{\text{best}}$ — the best variant, which is based on formulation (F2), with VCI and SSI separation, and with branching on variables u.

BCP$_{\text{best}-u}$ — the best variant without branching on variables u.

BCP$_{\text{best-VCI}}$ — the best variant without separating VCI.

BCP$_{\text{best-SSI}}$ — the best variant without separating SSI.

BCP$_{\text{best-(F1)}}$ — the best variant, but based on formulation (F1).

Table 2.2 gives the comparison of the BCP variants. It contains average values for the root gap, geometric mean values for the root solution time, the number of branch-and-bound nodes, the geometric mean of total solution time in seconds, and the number of instances solved within the time limit set to 3 hours. For unsolved instances, the solution time is set to the time limit.

Table 2.2: Comparison of variants of the BCP algorithm

	Root				
Variant	Gap (%)	Time (s)	Nodes	Time (s)	Solved
BCP$_0$	4.29	83.6	76.0	2333.0	31/54
BCP$_{+u}$	4.28	99.5	24.5	1020.5	44/54
BCP$_{\text{best-(F1)}}$	0.68	177.8	5.9	421.4	49/54
BCP$_{\text{best-SSI}}$	1.64	115.5	12.5	426.1	49/54
BCP$_{\text{best-VCI}}$	0.71	238.6	6.6	501.0	50/54
BCP$_{\text{best}-u}$	0.67	161.2	7.0	384.7	50/54
BCP$_{\text{best}}$	0.68	159.0	6.3	361.7	51/54

We see that the base variant BCP$_0$ is the worst as it solves only 31 out of 54 instances. Adding branching on variables u improves significantly BCP$_0$ as variant BCP$_{+u}$ solves 13 more instances. The best variant BCP$_{\text{best}}$ solves 7 more instances to optimality within

the time limit. Table 2.2 shows that all our contributions improve the efficiency of the BCP algorithm. The root gap decreases significantly when VCI and SSI are separated. Although the root solution time increases when additional inequalities are separated, the overall time decreases due to the much smaller size of the branch-and-bound tree. Thus, branching on variables u has a small effect on the performance of BCP_{best}. However, adding branching on variables u is the simplest way to make BCP_0 much more efficient.

2.5.3 Comparison with the state-of-the-art algorithm

Let us now compare BCP_0 and BCP_{best} to the best exact algorithm by Baldacci et al. (2013). For a fair comparison, we do not use initial primal bounds in this experiment. Instead, we rely on the primal heuristic presented in section 2.4.4 to find feasible solutions. Baldacci et al. (2013) did not set an overall time limit for their algorithm. They set a limit on the number of collections of first-level routes considered, as well as a time limit of 5000 seconds for solving each subproblem with fixed first-level routes. In our BCP algorithm, we set the time limit to 10 hours.

Table 2.3 shows the summary results of this experiment. For each set of literature instances, we give the average gap between the root dual bound and the best primal bound found (Rg), the geometric mean of the number of branch-and-bound nodes (Nds), the geometric mean of the solution time in seconds (t), and the number of instances solved to optimality (Solved). For a fair comparison, the solution time of Baldacci et al. (2013) is divided by 1.6 because of the difference in computer speeds. The algorithm in Baldacci et al. (2013) was tested only on 6 out of 18 instances of set 5. It has not been applied for instances with 10 satellites, as it is based on an enumeration of subsets of first-level routes. Since our algorithms are free from this drawback, they were tested on all instances of set 5.

Table 2.3: Comparison of two BCP variants with the state-of-the-art exact algorithm for the 2E-CVRP (Baldacci et al., 2013)

	BCP_0				BCP_{best}				Literature	
Set	Rg(%)	Nds	t (s)	Solved	Rg(%)	Nds	t (s)	Solved	t (s)	Solved
4A	5.76	14.2	772	51/54	0.91	3.3	144	54/54	271	50/54
4B	4.45	12.7	550	52/54	0.98	3.6	203	54/54	232	52/54
5	5.83	222.9	20612	6/18	1.41	22.5	3215	15/18	8405	3/6
6A	7.04	99.7	2604	24/27	0.89	4.9	233	27/27	802	22/27
6B	3.15	57.8	1562	24/27	0.46	4.3	196	27/27	513	19/27

The variant BCP_0 solves to optimality more instances than the best algorithm in the literature. However, the running time of the latter is on average smaller. The variant

BCP_{best} largely outperforms both other algorithms for all sets of instances. Indeed, BCP_{best} solves to optimality 31 open instance within 10 hours. The remaining 3 open instances were solved to optimality by providing the best-known solution of the literature as initial primal bound and using a special parameterisation.

Table 2.4 shows that our BCP algorithm could improve 10 best-known solutions (BKS) for literature instances. Their optimum solution values are given in column Opt. The improvement (Imp) in general is small. Thus, the existing heuristics for the 2E-CVRP have very good quality (at least when applied to literature instances).

Instance		BKS	Reference	Opt	Imp (%)
Set 5	100-5-1b	1103.55	Amarouche et al. (2018)	1099.35	0.38
	100-10-3b	849.73	Amarouche et al. (2018)	848.16	0.19
	200-10-1	1538.35	Amarouche et al. (2018)	1537.52	0.05
	200-10-1b	1175.81	Amarouche et al. (2018)	1173.07	0.23
	200-10-3	1779.68	Amarouche et al. (2018)	1177.49	0.12
	200-10-3b	1196.93	Amarouche et al. (2018)	1192.35	0.38
Set 6A	C-n101-4	1297.42	Wang et al. (2017)	1292.04	0.41
Set 6B	B-n101-4	1500.55	Breunig et al. (2016)	1499.71	0.06
	B-n101-5	1395.32	Breunig et al. (2016)	1394.79	0.04
	C-n101-5	1964.63	Breunig et al. (2016)	1962.52	0.11

Table 2.4: Improved best-known solutions for the 2E-CVRP literature instances

2.5.4 Experimental results for new instances

We tested the variant BCP_{best} on the set of newly generated instances involving 5 – 15 satellites and 100 – 300 customers. We set the time limit to 60 hours. We gave more time to the primal heuristic when solving the largest instances. For instances with 300 customers and 10 satellites, this time was set to 600 seconds. For instances with 15 satellites, this time was set to $4 \cdot |\mathcal{C}|$ seconds.

Out of 51 instances, our algorithm solved to optimality 23 instances, including some instances with 300 customers or with 15 satellites. The algorithm found both dual and primal bounds for 17 instances. The primal heuristic did not find any feasible solution for 9 instances having 300 customers and/or 15 satellites. Only lower bounds are thus currently known for these instances. We could not obtain dual bounds for 2 instances because the LP solver spent more than one hour to solve the restricted master LP during the first column generation convergence.

The main goal of this experiment was to generate instances which our best algorithm cannot solve in a reasonable time. This goal is achieved.

Chapter 3

Two-Echelon Vehicle Routing Problem with Time-Windows

The strong growth of home delivery services and e-commerce leads to a massive flow of goods to the city centers. This tends to bring trucks within cities, while the latter restrict or ban the use of polluting and large freight vehicles in city centers. In order to find alternative solutions to direct deliveries from distribution centers to customers, multi-echelon distribution networks were proposed by Crainic et al. (2009). The two-echelon distribution system is the simplest structure of these distribution networks. Trucks circulate in the first level outside the city center while small and clean vehicles are used in the second level for last-mile delivery. Light electric freight vehicles or cargo bikes as commonly used at this level since they are agile, quiet, emission-free, and takes up less space than vans or trucks. The connection between the two levels is ensured by intermediate warehouses such as Urban Consolidation Centers (UCC) (Allen et al., 2012), which provide temporary storage and consolidate the parcels flow in the last mile (McDermott, 1975). As the costs of these UCCs are high (Holguín-Veras et al., 2018), an alternative is to use intermediate warehouses with limited storage space or no storage at all. These warehouses called satellites are commonly based on existing infrastructure such as car parks, bus depots, or some street sidewalks. In this context, synchronization of flows at intermediate warehouses is therefore an essential feature in urban freight transport: exact synchronization constraints are encountered in satellites, and precedence constraints are encountered in UCCs. Time windows at customer sites are also commonly used in practice.

In this chapter, we study the two-echelon vehicle routing problem with time windows (2E-VRPTW), which consists in determining the number of goods to be shipped from the distribution centers to the satellites and from satellites to customers, together with

the optimal routes connecting entities in each level, such that vehicle capacities are not exceeded, customer demands are satisfied, customers are delivered within their time windows, and first-level routes precede second-level routes to do transfers at satellites. The goal is to minimize operational and transportation costs.

Two-echelon vehicle routing with time windows has received little attention in the literature so far. Usually, in the variants of the problem considered in the literature, exact synchronization is required, and one forbids freight consolidation, i.e. the loading to a city freighter from several urban trucks. Such constraints are imposed, for example, in the papers by Grangier et al. (2016) and Dellaert et al. (2019). In contrast to these papers, our problem variant allows for consolidation and does not require exact synchronization for transfers because we use precedence constraints at satellites. Our case is thus suited for practical situations with UCCs, i.e. satellites with relatively large storage capacities. In this work, we propose the first exact algorithm for this variant of the problem. The algorithm is based on the branch-cut-and-price (BCP) approach.

We would like to underline that our algorithm is useful both for the case with precedence constraints and for the case with exact synchronization. Indeed, exact algorithms are generally used in practice to obtain valid lower bounds on the value of an optimum solution to the problem. These bounds are then used to estimate the quality of heuristic algorithms. As the variant of the 2E-VRPTW with precedence constraints and consolidation is a relaxation for the variant with exact synchronization, the lower bounds obtained by our algorithm are valid for both cases. Moreover, we provide a post-processing procedure that allows one to minimize the usage of storage in a given solution without increasing its transportation cost.

We now summarize the main contributions of our work presented in this chapter.

- We introduce a new mixed-integer programming (MIP) formulation for the 2E-VRPTW with precedence constraints and freight consolidation. This formulation involves an exponential number of route variables and an exponential number of precedence constraints. Our formulation does not involve variables which explicitly model freight transfer at satellites. This fact greatly simplifies the following approach to solve the formulation.

- To solve the introduced formulation to optimality, we propose a branch-cut-and-price algorithm, which combines column and cut generation with strong branching and an enumeration procedure for elementary routes (Baldacci et al., 2008b). Our algorithm incorporates many advanced techniques proposed recently for tackling classic vehicle routing problems. It includes an original separation algorithm that

generates violated precedence constraints. We also show how precedence constraints can be taken into account when solving the pricing problem in column generation.

- We show how to adapt our BCP algorithm for a more practically relevant variant of the problem, in which city freighters can perform multiple trips.
- We perform extensive computational evaluation of our algorithm using literature instances introduced by Grangier et al. (2016) and Dellaert et al. (2019). Experiments reveal that it can solve to optimality single-trip instances with up to 6 distribution centers, 5 satellites and 100 customers, and multi-trip instances with up to 8 satellites and 100 customers. Moreover, we show that: (i) virtually all instances proposed by Dellaert et al. (2019) have optimum solutions with the same transportation cost for both variants of the problem with precedence constraints and with exact synchronization; (ii) our algorithm solves to optimality 54 open instances of the single-trip 2E-VRPTW; (iii) it outperforms significantly the algorithm proposed by Dellaert et al. (2019) on their instances.

The remaining of the paper is organized as follows. Section 3.1 reviews the literature. MIP formulations of the problem are introduced in Section 3.2. In Section 3.3, we describe the proposed branch-cut-and-price algorithm. In Section 3.4, we present and discuss the computational results. The appendices with detailed results will be available with the published paper.

3.1 Literature review

The 2E-VRPTW is a generalization of the quite well-studied two-echelon capacitated vehicle routing problem (2E-CVRP). Several exact approaches have been proposed for the 2E-CVRP. Branch-and-cut algorithms were suggested by Gonzalez-Feliu et al. (2007); Perboli et al. (2011); Jepsen et al. (2013); and Contardo et al. (2012). An exact method based on the enumeration of first-level solutions was proposed by Baldacci et al. (2013). The first branch-cut-and-price algorithm was developed by Santos et al. (2015). Recently, Marques et al. (2020) published an improved branch-cut-and-price algorithm which outperforms other exact methods in the literature. Optimum solutions can now be consistently obtained for instances with up to 200 customers and 10 satellites.

Several heuristic approaches for the 2E-CVRP have been proposed in the literature. A large neighbourhood search-based method has been suggested by Hemmelmayr et al. (2012b) and by Breunig et al. (2016). Zeng et al. (2014) proposed a hybrid heuristic that combines greedy randomized adaptive search procedure and a variable neighbourhood

descent. Matheuristics that combine local search to build routes and a route-based MIP to derive complete solutions were employed by Wang et al. (2017) and Amarouche et al. (2018). The latter two algorithms are the best heuristics for the problem available in the literature until today. More details on the 2E-CVRP can be found in the survey paper by Cuda et al. (2015).

The 2E-VRPTW and its variants are less studied than the 2E-CVRP, although the former is more relevant in practice. Grangier et al. (2016) suggested a mathematical formulation of the variant of the 2E-VRPTW with multiple trips and exact synchronization (MT-2E-VRPTW-ES), in which freight consolidation is forbidden. A city freighter thus receives products from only one urban truck. The authors proposed an adaptive large neighbourhood search heuristic that embeds customized destroy and repair procedures. Their objective function successively minimizes the number of urban trucks used, the number of city freighters used, and the total distance traveled. They experimented with instances involving 8 satellites and 100 customers and searched for feasible solutions within two hours.

Dellaert et al. (2019) suggested three MIP formulations for the single-trip 2E-VRPTW with exact synchronization (ST-2E-VRPTW-ES), in which freight consolidation is also forbidden. First, they introduced an arc-based formulation optimized using a commercial MIP solver. This approach could only solve instances with 15 customers within one hour. Secondly, they proposed a "tour-tree" based formulation, in which a tour-tree is a combination of a first-level route and the second-level routes loaded by this first-level route. A branch-and-price algorithm was devised to tackle this formulation. Again, only instances with up to 15 customers could be solved. Finally, the authors proposed a route based formulation and an enumeration-based algorithm similar to the one by Baldacci et al. (2013) to solve the ST-2E-VRPTW-ES. The method generates collections of first-level routes, then iteratively fixes the first-level routes by choosing the most promising collection according to a lower bound, and finally optimizes the second-level problem as a multi-depot vehicle routing problem with time windows using a branch-and-price algorithm. This method could solve most instances with up to 50 customers and some instances with 100 customers.

Li et al. (2016) considered a variant of the 2E-VRPTW for linehaul delivery systems with exact synchronization. They suggested a MIP formulation and tackled the problem with combination of an initial Clarke-and-Wright savings heuristic and a local search. Li et al. (2020b) studied another variant of the 2E-VRPTW with mobile satellites. The first echelon involves vans and the second echelon involves unmanned aerial vehicles (UAV). The vans parked at some customer locations are used as mobile satellites from which

drones deliver other customers. They proposed a vehicle-flow formulation which involves non-exact synchronization constraints at mobile satellites and time windows at customer locations. Given the specific nature of the distribution with UAVs at the second level, their model is not dedicated to achieve freight consolidation at satellites. An adaptive large neighbourhood search (ALNS) heuristic has also been proposed, which were tested on instances with up to 100 customers derived from the standard VRPTW benchmark. Li et al. (2020a) considered a city logistics distribution system with on-street satellites and time windows at customer locations, which has some similarities with the 2E-VRPTW. Synchronization is performed at satellites where freight consolidation takes place. The problem is formulated as a MIP, and it is optimized with a variable neighbourhood search (VNS) heuristic. The latter was tested on instances with up to 30 on-street-satellites and 900 customers. Nolz et al. (2020) considered two-echelon urban distribution systems with a single capacitated city hub and exact synchronization between echelons. For this setting, they proposed a three-phase heuristic method which uses population-based meta-heuristics and integer programs.

Related works include models developed for certain specific application areas. Wang and Wen (2020) focused on a variant of the 2E-VRPTW with soft time windows and a heterogeneous fleet of vehicles for the cold chain logistics. They proposed an adaptive genetic algorithm and optimized small instances with 2 distribution centers, 15 customers, and 3 satellites. He and Li (2019) proposed a memetic algorithm for a multi-trip variant of the 2E-VRPTW arising in agriculture, where second-level harvesting machines have to visit many farmlands and unload the crop at one or many of them (i.e. satellites) into first-level trucks. In this problem, the satellite location is changed continuously during the working day and a non-exact synchronization between the two levels is defined by a time window at each satellite. There is no consolidation in this model. The authors used a set of instances with up to 250 customers.

To conclude, several variants of the 2E-VRPTW have been studied in the literature in recent years. Nevertheless, to our knowledge, only one exact method has been proposed so far by Dellaert et al. (2019) to the 2E-VRPTW. In this chapter, we address the lack of exact methods by proposing an algorithm that has broader applicability than the existing one.

3.2 Problem definition and formulation

We now formally define the problem. At the first level, a fleet $\mathcal{U}$ of homogeneous urban trucks ships goods from a set $\mathcal{D}$ of distribution centers to a set $\mathcal{S}$ of satellites. The capacity

of an urban truck is Q_1 items. The tour of an urban truck starts at a distribution center, delivers freight to some satellites, and ends at the same distribution center. The second level involves a set $\mathcal{F}$ of homogeneous city freighters that ship freight from satellites to a set $\mathcal{C}$ of customers. The capacity of a city freighter is Q_2 items. Each customer $c \in \mathcal{C}$ has an integer demand d_c and must be visited by one city freighter. The latter must arrive at the location of customer $c \in \mathcal{C}$ within a time window starting at time l_c and ending at time u_c (waiting is possible for early arrival). Once arrived, the city freighter needs σ_c time units to serve the customer. In the single-trip variant, a city freighter starts its tour from a satellite, visits some customers, and ends at the same satellite. In the multi-trip variant, a city freighter starts from a unique depot, goes to a satellite, delivers some customers, and goes empty to a satellite to start another trip or ends at the depot.

Transfers of freight from urban trucks to city freighters take place at satellites. Vehicles can arrive at satellite $s \in \mathcal{S}$ within a time window $[l_s, u_s]$. In our variant, the freight consolidation is allowed. This means that a city freighter can receive freight from several urban trucks. Moreover, the exact synchronization of an urban truck and a city freighter at a satellite is not required. A transfer at satellite $s \in \mathcal{S}$ consists of the following steps. An urban truck arrives at the satellite, possibly waits until the beginning of time window l_s, stays during service time σ_s, and then leaves. A city freighter arrives at the satellite, possibly waits until the start of service time of an urban truck from which the city freighter gets its freight, stays during service time σ_s, and then leaves. Figure 3.1 depicts examples of feasible transfers.

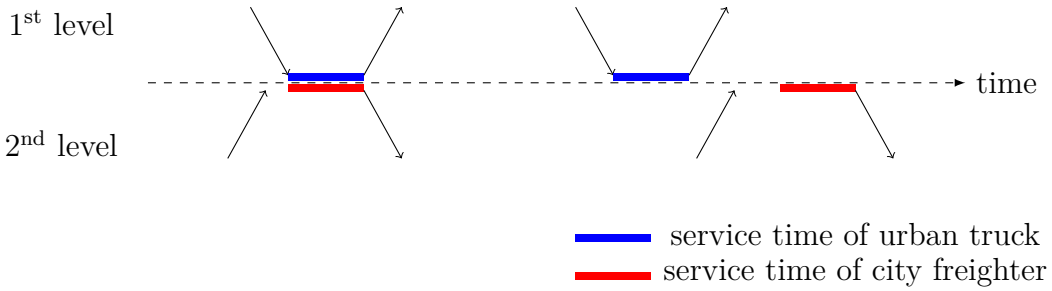


Figure 3.1: Examples of transfers at a satellite

For the sake of clarity, we now focus on the single-trip variant of the problem. Specificities of the multi-trip variant are discussed in Section 3.2.4.

The first-level problem is similar to the split-delivery CVRP because several urban trucks can supply a satellite. However, the amount of freight delivered to each satellite is not fixed. The second level problem is similar to the multi-depot CVRP with time windows, in which satellites take the role of depots.

The distribution system is represented by two graphs. Directed graph $G_1 = (V_1, A_1)$ with $V_1 = \mathcal{D} \cup \mathcal{S}$ and $A_1 = \mathcal{D} \times \mathcal{S} \cup \{(s, s') \in \mathcal{S}^2 : s \neq s'\} \cup \mathcal{S} \times \mathcal{D}$ represents the first level of the distribution system. Directed graph $G_2 = (V_2, A_2)$ with $V_2 = \mathcal{S} \cup \mathcal{C}$ and $A_2 = \mathcal{S} \times \mathcal{C} \cup \{(c, c') \in \mathcal{C}^2 : c \neq c'\} \cup \mathcal{C} \times \mathcal{S}$ represents the second level of the distribution system. For each arc $a \in A_1 \cup A_2$, travelling cost f_a and travel time t_a are given.

We denote as P the set of feasible first-level routes. A route $p \in P$ is an elementary cycle $(v_0^p, v_1^p, \ldots, v_{n(p)}^p)$ in G_1, in which $v_0^p = v_{n(p)}^p \in \mathcal{D}$, and $v_k^p \in \mathcal{S}$, $1 \leq k < n(p)$. We denote as S_p the set of satellites visited by route $p \in P$: $S_p = \{v_1^p, \ldots, v_{n(p)-1}^p\} \subseteq \mathcal{S}$. Since our variant allows for storage of items at satellites, there exists an optimal solution in which each first-level route visits each satellite at most once and departs from each node as early as possible. Let $\tilde{f}^p$ denote the cost of route $p \in P$, which includes the total travel cost and the fixed cost of using an urban truck. Also let $\tilde{t}_k^p$ denote the departure time of route $p \in P$ from node v_k^p, $0 \leq k \leq n(p)$. Without loss of generality, each value $\tilde{t}_k^p$ can be fixed to the earliest departure time:

$$\tilde{t}_k^p = \begin{cases} \sigma_{v_k^p}, & k = 0, \\ \max\left\{\tilde{t}_{k-1}^p + t_{(v_{k-1}^p, v_k^p)},\ l_{v_k^p}\right\} + \sigma_{v_k^p}, & 1 \leq k \leq n(p). \end{cases}$$

A first-level route p is feasible if $\tilde{t}_k^p \leq u_{v_k^p} + \sigma_{v_k^p}$, $1 \leq k \leq n(p)$. For a pair (s, t), where $s \in \mathcal{S}$ and $0 \leq t \leq u_s$, we denote as P_{st}, the set of first-level routes which visit satellite s and depart from it strictly before time moment t: $P_{st} = \{p \in P : \exists k, 1 \leq k < n(p), v_k^p = s, \tilde{t}_k^p < t\}$.

We denote as R_s the set of feasible second-level routes starting from satellite s. Let also $R = \bigcup_{s \in \mathcal{S}} R_s$. A route $r \in R_s$ is an elementary cycle $(v_0^r, v_1^r, \ldots, v_{n(r)}^r)$ in G_2, in which $v_0^r = v_{n(r)}^r = s$, and $v_k^r \in \mathcal{C}$, $1 \leq k < n(r)$. Again, since our variant allows for storage of items at satellites, there exists an optimal solution in which each second-level route departs from each node as late as possible. Let $\tilde{z}_c^r$ be equal to 1 if route $r \in R$ serves customer $c \in \mathcal{C}$, and 0 otherwise. Let $\tilde{d}^r$ be the total amount of freight delivered by route $r \in R$: $\tilde{d}^r = \sum_{c \in \mathcal{C}} d_c \tilde{z}_c^r \leq Q_2$. Let $\tilde{f}^r$ denote the cost of route $r \in R$, which includes the total travel cost and the fixed cost of using a city freighter. Also let $\tilde{t}_k^r$ denote the departure time of route $r \in R$ from node v_k^p, $0 \leq k \leq n(p)$. Without loss of generality, each value $\tilde{t}_k^r$ can be fixed to the latest departure time:

$$\tilde{t}_k^r = \begin{cases} u_{v_k^r}, & k = n(r), \\ \min\left\{\tilde{t}_{k+1}^r - \sigma_{v_{k+1}^r} - t_{(v_k^r, v_{k+1}^r)},\ u_{v_k^r} + \sigma_{v_k^r}\right\}, & 0 \leq k < n(r). \end{cases}$$

A second-level route r is feasible if $\tilde{t}^r_k \geq l_{v^r_k} + \sigma_{v^r_k}$, $0 \leq k < n(r)$. For a pair (s,t), where $s \in \mathcal{S}$ and $0 \leq t \leq u_s$, we denote as R_{st} the set of second-level routes in R_s which depart from satellite s strictly before time moment t: $R_{st} = \{r \in R_s : \tilde{t}^r_0 < t\}$.

A feasible solution to the problem consists of a set of feasible first-level and second-level routes satisfying the following partitioning, precedence, and capacity constraints:

(C1) each customer is visited by exactly one second-level route,

(C2) for each satellite $s \in S$ and each time moment $0 \leq t \leq u_s$, the total amount of freight, delivered to s by first-level routes in P_{st}, is not smaller than the total amount of freight delivered by second-level routes in R_{st},

(C3) the total amount of freight delivered by every first-level route does not exceed Q_1.

The objective function is the same as the one used by Dellaert et al. (2019): we need to minimize the sum of the total travelling cost and the total fixed cost of vehicles usage, i.e. the total routes cost.

3.2.1 Standard formulation

Let integer variable λ_p, $p \in P$, be equal to the number of urban trucks which follow first-level route p. Let binary variable μ_r, $r \in R$, takes value 1 if a city freighter follows second-level route r, and 0 otherwise. Let continuous variable w_{ps}, $p \in P$, $s \in S_p$, be equal to the amount of freight that first-level route p delivers to satellite s. Then our problem can be formulated as follows.

$$\text{(F1)} \quad \min \quad \sum_{p \in P} \tilde{f}^p \lambda_p + \sum_{r \in R} \tilde{f}^r \mu_r \tag{3.1}$$

$$\text{s.t.} \quad \sum_{r \in R} \tilde{z}^r_c \mu_r = 1 \qquad c \in \mathcal{C} \tag{3.2}$$

$$\sum_{p \in P_{st}} w_{ps} - \sum_{r \in R_{st}} \tilde{d}^r \mu_r \geq 0 \qquad s \in \mathcal{S},\ l_s < t \leq u_s, \tag{3.3}$$

$$\sum_{s \in S_p} w_{ps} \leq Q_1 \lambda_p \qquad p \in P \tag{3.4}$$

$$\lambda_p \in \mathbb{Z}_+ \qquad p \in P \tag{3.5}$$

$$\mu_r \in \{0,1\} \qquad r \in R \tag{3.6}$$

$$w_{ps} \geq 0 \qquad p \in P,\ s \in S_p \tag{3.7}$$

The objective function (3.1) minimizes the total routes cost. Partitioning constraints (3.2), precedence constraints (3.3), and capacity constraints (3.4) correspond to

constraints (C1), (C2), and (C3) respectively. Constraints (3.5), (3.6) and (3.7) define the domains of variables. The number of precedence constraints (3.3) can be reduced to a finite number while keeping the formulation valid. We define as T_s the set of all time moments at which first-level routes leave satellite s: $T_s = \left\{\tilde{t}^p_k : p \in P,\ 1 \le k < n(p),\ v^p_k = s\right\}$. Then it suffices to keep only constraints (3.3) for pairs (s,t) such that $s \in \mathcal{S}$ and $t \in T_s$.

Formulation (F1) cannot be solved directly in practice as the number of variables and constraints is very large. Even the standard column and cut generation approach is not suited to solve its linear programming (LP) relaxation. This is because for every newly generated variable λ_p, $p \in P$, one should also generate variables w_{ps}, $s \in S_p$, and the corresponding constraint (3.4).

3.2.2 Modified formulation

In this section, we modify formulation (F1) so that dynamic generation of route variables does not require simultaneous generation of constraints. For the modified formulation, we are able to compute the current reduced cost of route variables. The standard column generation procedure then can be used once the restricted set of precedence constraints is fixed. Therefore, the precedence constraints separation procedure may alternate with the column generation procedure. The overall columns and cut generation procedure stops when no negative reduced columns are found and no precedence constraints are violated.

The main idea of the modified formulation is to merge constraints (3.3) and (3.4) to remove variables w. First we need to introduce some notation. Given a time vector $\tau = (\tau_s)_{s\in\mathcal{S}}$, let $P(\tau)$ be the set of first-level routes which depart from a satellite $s \in \mathcal{S}$ before time τ_s: $P(\tau) = \{p \in P : \exists k,\ 1 \le k < n(p),\ \tilde{t}^p_k < \tau_{v^p_k}\}$. Analogously, let $R(\tau)$ be the set of second-level routes which depart from a satellite $s \in \mathcal{S}$ before time τ_s: $R(\tau) = \{r \in R : \tilde{t}^r_0 < \tau_{v^r_0}\}$. Also, let $\mathcal{T}$ be the cartesian product of all sets T_s, $s \in \mathcal{S}$, extended by value 0, i.e. $\mathcal{T} = \bigtimes_{s\in\mathcal{S}}(T_s \cup \{0\})$. The modified formulation is then the following.

$$\text{(F2)}\quad \min \quad \sum_{p\in P} \tilde{f}^p \lambda_p + \sum_{r\in R} \tilde{f}^r \mu_r \tag{3.8}$$

$$\text{s.t.} \quad \sum_{r\in R} \tilde{z}^r_c \mu_r = 1 \qquad c \in \mathcal{C} \tag{3.9}$$

$$\sum_{p\in P(\tau)} Q_1 \lambda_p - \sum_{r\in R(\tau)} \tilde{d}^r \mu_r \ge 0 \qquad \tau \in \mathcal{T} \tag{3.10}$$

$$\lambda_p \in \mathbb{Z}_+ \qquad p \in P \tag{3.11}$$

$$\mu_r \in \{0,1\} \qquad r \in R \tag{3.12}$$

We call constraints (3.10) two-level precedence constraints (TLPC). They replace constraints (3.3) and (3.4). Before showing that formulation (F2) is equivalent to (F1), we give an example of a violated TLPC.

Example 6. *Consider an instance of 2E-VRPTW with one distribution center, three satellites* $\mathcal{S} = \{s_1, s_2, s_3\}$*, and a set of customers that has a total demand of* 55 *items. Capacities of vehicles are* $Q_1 = 20$ *and* $Q_2 = 13$*. Suppose we are given the solution depicted by Figure 3.2. Urban truck following route* p_1 *takes* 20 *units of freight from the distribution center and delivers them to satellite* s_2 *at time* 5 *and satellite* s_1 *at time* 15*. Urban truck taking route* p_2 *delivers* 20 *items to* s_2 *at time* 55*. Urban truck taking route* p_3 *delivers* 15 *items to* s_1 *at time* 75*. City freighters taking routes* r_1 *and* r_5 *start from satellite* s_1 *at time moments* 27 *and* 105 *with loads of* 10 *and* 13 *items respectively. City freighters taking routes* r_2*,* r_3*, and* r_4 *start from* s_2 *at time moments* 40*,* 63*, and* 85 *with loads of* 12*,* 7*,* 13 *items respectively.*

We now consider the TLPC characterized by time vector $\tau = (70, 50, 0)$*. This TLPC involves routes* p_1*,* r_1*, and* r_2 *arriving and leaving satellites in the gray area in Figure 3.2. Since* p_1 *delivers* 20 *items and* r_1*, and* r_2 *cover* 22 *items of demand, the solution violates this TLPC.*

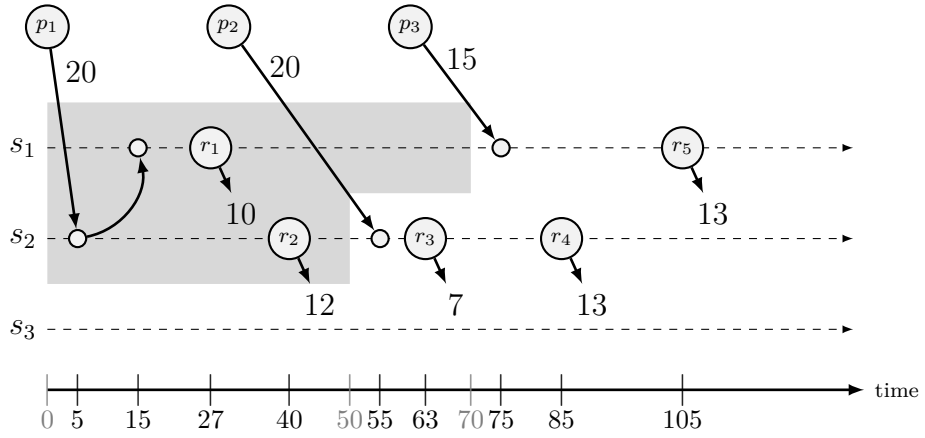


Figure 3.2: Example of a solution and a violated TLPC characterized by vector $\tau = (70, 50, 0)$

We now prove that (F2) is a projection of (F1). The proof is illustrated in Figure 3.3.

Proposition 5. *A solution* $(\bar{\lambda}, \bar{\mu})$ *is feasible to the LP relaxation (LF2) of formulation (F2) if and only if there exists a feasible solution* $(\bar{\lambda}, \bar{\mu}, \bar{w})$ *to the LP relaxation (LF1) of formulation (F1).*

Proof. Proof. To prove sufficiency ("if" part), we need to show that constraints (3.10) are valid for (LF1). Let fix a feasible solution $(\bar{\lambda}, \bar{\mu}, \bar{w})$ to (LF1). For an arbitrary time vector $\tau \in \mathcal{T}$, we have

$$\sum_{r \in R(\tau)} \tilde{d}^r \bar{\mu}_r = \sum_{s \in \mathcal{S}} \sum_{r \in R_{s,\tau_s}} \tilde{d}^r \bar{\mu}_r \overset{(3.3)}{\leq} \sum_{s \in \mathcal{S}} \sum_{p \in P_{s,\tau_s}} \bar{w}_{ps} \leq \sum_{p \in P(\tau)} \sum_{s \in S_p} \bar{w}_{ps} \overset{(3.4)}{\leq} \sum_{p \in P(\tau)} Q_1 \bar{\lambda}_p$$

Thus, $(\bar{\lambda}, \bar{\mu})$ is feasible to (LF2).

We now prove necessity, i.e. "only if" part. Consider a feasible solution $(\bar{\lambda}, \bar{\mu})$ to (LF2). We denote as $\bar{P}$ and $\bar{R}$ the sets of first-level and second-level routes participating in the solution: $\bar{P} = \{p \in P : \ \bar{\lambda}_p > 0\}$ and $\bar{R} = \{r \in R : \ \bar{\mu}_r > 0\}$.

We build the directed graph $\bar{G} = (\bar{V}, \bar{A})$. The set of nodes is $\bar{V} = \{\bar{s}, \bar{t}\} \cup \bar{P} \cup \bar{R}$, where $\bar{s}$ is the source, and $\bar{t}$ is the sink. Set $\bar{A}$ of arcs consists of three subsets. Subset $\bar{A}_1$ contains, for each $p \in \bar{P}$, arc $(\bar{s}, p)$ with capacity $Q_1 \bar{\lambda}_p$. Subset $\bar{A}_2$ contains arc $(p, r) \in \bar{P} \times \bar{R}$ if and only if first level route p leaves satellite $s = v_0^r$ before or at the same time as second-level route r. Every arc in $\bar{A}_2$ has infinite capacity. Subset $\bar{A}_3$ contains, for each $r \in \bar{R}$, arc $(r, \bar{t})$ with capacity $\tilde{d}^r \bar{\mu}_r$.

Let us now prove by contradiction that the maximum flow value from $\bar{s}$ to $\bar{t}$ in $\bar{G}$ is equal to $\sum_{c \in \mathcal{C}} d_c = d(\mathcal{C})$. Assume that the maximum flow value is strictly less than $d(\mathcal{C})$. Let $\bar{V}'$ be the subset of $\bar{V}$ obtained from a minimum $\bar{s}$-$\bar{t}$ cut in $\bar{G}$, where $\bar{s} \in \bar{V}'$. Let $\bar{P}' = \bar{P} \setminus \bar{V}'$ and $\bar{R}' = \bar{R} \setminus \bar{V}'$. Let $\bar{A}'$ be the set of arcs that cross the cut: $\bar{A}' = \{(i, j) \in \bar{A} : \ i \in \bar{V}', \ j \notin \bar{V}'\}$. From the assumption and the max-flow-min-cut theorem, it follows that the total capacity of arcs in $\bar{A}'$ is less than $d(\mathcal{C})$. Thus $\bar{A}'$ does not contain all arcs in $\bar{A}_3$ and $\bar{A}'$ contains at least one arc in $\bar{A}_1$. Therefore, the total capacity of arcs in $\bar{A}_1 \cap \bar{A}'$ is strictly less than the total capacity of arcs in $\bar{A}_3 \setminus \bar{A}'$:

$$\sum_{p \in \bar{P}'} Q_1 \bar{\lambda}_p < \sum_{r \in \bar{R}'} \tilde{d}^r \bar{\mu}_r. \tag{3.13}$$

Consider now time vector $\bar{\tau}$ such that

$$\bar{\tau}_s = \begin{cases} \epsilon + \max_{r \in \bar{R}' \cap R_s} \{\tilde{t}_0^r\}, & \bar{R}' \cap R_s \neq \emptyset, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall s \in \mathcal{S}. \tag{3.14}$$

Here ϵ is a very small positive value. No first-level route $\bar{P} \setminus \bar{P}'$ can serve second-level route in $\bar{R}'$, as otherwise the minumum $\bar{s}$-$\bar{t}$ cut would have infinite value. Thus, set $P(\bar{\tau})$

does not contain any route in $\bar{P} \setminus \bar{P}'$, and

$$\sum_{p \in P(\bar{\tau})} Q_1 \bar{\lambda}_p = \sum_{p \in \bar{P}'} Q_1 \bar{\lambda}_p \overset{(3.13)}{<} \sum_{r \in \bar{R}'} \tilde{d}^r \bar{\mu}_r \leq \sum_{r \in R(\bar{\tau})} \tilde{d}^r \bar{\mu}_r.$$

Thus, solution $(\bar{\lambda}, \bar{\mu})$ violates the precedence constraint (3.10) characterized by time vector $\bar{\tau}$ and that contradicts the fact that this solution is feasible to (LF2). Then, our assumption about the maximum flow value is wrong, and this value is equal to $d(\mathcal{C})$. We now set each value $\bar{w}_{ps}$, $p \in P$, $s \in \mathcal{S}$, equal to the total flow value along all arcs (p, r) in $\bar{A}_2$ such that $r \in R_s$, and to 0 if there are no such arcs. By construction of graph $\bar{G}$, constraints (3.3) and (3.4) are satisfied by solution $(\bar{\lambda}, \bar{\mu}, \bar{w})$, and the latter is feasible to (LF1). □

□

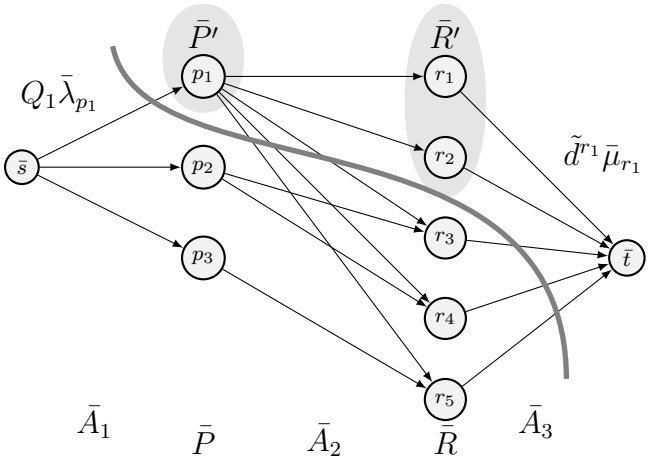


Figure 3.3: Minimum cut in graph $\bar{G}$ based on the solution in Example 6.

The linear relaxation (LF2) is solved by the column and cut generation procedure described in Section 3.3. To improve the lower bound for the 2E-VRPTW obtained by this procedure, we use four families of valid inequalities, described in the next section.

3.2.3 Valid inequalities

We use the following families of valid inequalities introduced in the two previous chapters : RCCs (1.25), lm-R1Cs (1.26), VCIs (2.15), and SSIs (2.22).

3.2.4 Multi-trip variant

In the multi-trip variant, the first level of the distribution system stays the same but the second level changes because city freighters can perform several trips and can visit more than one satellite. The second level problem becomes the multi-depot multi-trip CVRP with time windowsand it is similar to the multi-depot VRP with interdepot routes considered by (Muter et al., 2014). We consider two graph representations of the second level. In representation (R1) below, we do not keep track of the satellite from which the current trip started. In representation (R2), we keep track of the latest visited satellite. In the former, the second-level graph is smaller, but some valid inequalities described in Section 3.2.3 cannot be used.

(R1) The second level of the distribution system is represented by directed graph $G_2' = (V_2', A_2')$ where $V_2' = \{0\} \cup \mathcal{S} \cup \mathcal{C}$, $A_2' = A_2 \cup \{0\} \times \mathcal{S} \cup \mathcal{C} \times \{0\}$, and node 0 is the depot of city freighters. A trip in graph G_2' starts in a satellite $s \in \mathcal{S}$, visits some customers in $\mathcal{C}$, and goes empty to a satellite $s' \in \mathcal{S}$ (possibly $s = s'$) or to depot 0. For each arc $a \in \{0\} \times \mathcal{S} \cup \mathcal{C} \times \{0\}$, its travelling cost f_a and travel time t_a are given.

(R2) In this representation, each customer $c \in \mathcal{C}$ is represented by $|\mathcal{S}|$ nodes, one per satellite, instead of one. Let $\mathcal{C}_s$ be the set of customer nodes for satellite $s \in \mathcal{S}$, and let $\hat{\mathcal{C}} = \bigcup_{s\in\mathcal{S}} \mathcal{C}_s$. The second level is thus represented by directed graph $G_2'' = (V_2'', A_2'')$, where $V_2'' = \{0\} \cup \mathcal{S} \cup \hat{\mathcal{C}}$, $A_2'' = (\bigcup_{s\in\mathcal{S}} A_{2s}'') \cup \{0\} \times \mathcal{S}$, and $A_{2s}'' = \{s\} \times \mathcal{C}_s \cup \{(c, c') \in \mathcal{C}_s^2 : c \neq c'\} \cup \mathcal{C}_s \times \mathcal{S} \cup \mathcal{C}_s \times \{0\}$. We say that an arc $a'' \in A_2''$ projects to an arc $a' \in A_2'$ if their tails and heads correspond to the same satellite or customer. For an arc $a' \in A_2'$, Let $A''(a')$, be the set of arcs in A_2'' projecting to arc a'. A trip in graph G_2'' starts in a satellite $s \in \mathcal{S}$, visits some customers in $\mathcal{C}_s$, and goes empty to a satellite $s' \in \mathcal{S}$ (possibly $s = s'$) or to depot 0.

Let R' be the set of feasible multi-trip second-level routes. Each route consists of the first arc going from depot 0 to a satellite $s \in \mathcal{S}$, and one or several consecutive trips such that the first trip starts at s, each other trip starts at the satellite at which the previous trip has ended, and the last trip finishes at node 0. Let I_r, $r \in R'$, be the set of trips of a multi-trip route. As in the single-trip case, there exists an optimal solution in which each second-level route departs from each node as late as possible. We use the same notation R' for both graph representations (R1) and (R2), as there is a bijection between feasible routes in graphs G_2' and G_2''.

Let $\tilde{z}_c^i$ be equal to 1 if trip $i \in I_r$, $r \in R'$, serves customer $c \in \mathcal{C}$, and 0 otherwise. We have $\tilde{z}_c^r = \sum_{i \in I_r} \tilde{z}_c^i$ for a route $r \in R'$. Let $\tilde{y}_{a'}^r$ be equal to 1 if route $r \in R'$ uses arc $a' \in A_2'$, if representation (R1) is used, or uses an arc in $A''(a')$ if representation (R2) is used. Let $\tilde{d}^i$ be the total amount of freight delivered by trip $i \in I_r$, $r \in R'$: $\tilde{d}^i = \sum_{c \in \mathcal{C}} d_c \tilde{z}_c^i \leq Q_2$. Let $\tilde{t}^i$ and $\tilde{s}^i$ be the departure time of trip $i \in I_r$, and the satellite from which this trip departs.

We denote $I_{rs}(t)$ as the set of trips of route $r \in R'$ starting from satellite s before time t: $I_{rs}(t) = \{i \in I_r : \tilde{s}^i = s, \tilde{t}^i < t\}$. Given a time vector $\tau = (\tau_s)_{s \in \mathcal{S}}$ and a route $r \in R'$, let $I_r(\tau) = \cup_{s \in \mathcal{S}} I_{rs}(\tau_s)$. Then, two-level precedence constraints (3.10) can be rewritten for the multi-depot variant of the problem:

$$\sum_{p \in P(\tau)} Q_1 \lambda_p - \sum_{r \in R'} \sum_{i \in I_r(\tau)} \tilde{d}^i \mu_r \geq 0, \qquad \tau \in \mathcal{T}. \tag{3.15}$$

As values $\tilde{z}_c^r$, $c \in \mathcal{C}$, and $\tilde{y}_a^r$, $a \in A_2$, are defined for all multi-trip second-level routes, valid inequalities (1.25) and (1.26) can directly be used in the multi-trip case.

The branch-cut-and-price algorithm presented in the next section has two variants for the multi-trip case, depending on the graph representation used. If representation (R2) is used, for a given triple (r, s, a), $r \in R'$, $s \in \mathcal{S}$, $a \in A_2$, we are able to determine value $\tilde{y}_{sa}^r$, which is equal to 1 if an arc $a'' \in A_{2s}''$ projecting to a is used by route r. Then, variables $y_{sa} = \sum_{r \in R'} \tilde{y}_{sa}^r \mu_r$ are available, and we can use valid inequalities (2.22) and (2.15).

3.3 Branch-cut-and-price algorithm

To simplify presentation, we use additional auxiliary variables. Let $\tilde{x}_a^p$ be equal to 1 if route $p \in P$ uses arc $a \in A_1$, and 0 otherwise. Let $\tilde{y}_a^r$ be equal to 1 if route $r \in R$ uses arc $a \in A_2$, and 0 otherwise. Let integer variable ν_S, $S \subseteq \mathcal{S}$, be equal to the number of urban trucks visiting at least one satellite in S: $\nu_S = \sum_{p \in P:\, S_p \cap S \neq \emptyset} \lambda_p$.

The LP relaxation (LF2) of formulation (F2) together with valid inequalities (1.25), (1.26), (2.22), and (2.15) is solved by a column and cut generation approach. The first-level and second-level route variables are generated by solving the pricing problems which we describe in Section 3.3.1. We also show how two-level precedence constraints (3.10) affect the structure of the pricing problems. In Section 3.3.2, we introduce a separation algorithm for TLPC (3.10). We give a brief description of the remaining components of the branch-cut-and-price algorithm in Section 3.3.3. Finally, in Section 3.3.4, we present

the post-processing procedure that tries to exactly synchronize urban trucks and city freighters.

3.3.1 Pricing problems

Consider formulation (LF2) with a restricted number of variables and constraints. We denote it as (RLF2). Let $(\bar{\pi}, \bar{\zeta}, \bar{\rho}, \bar{\xi}, \bar{\eta}, \bar{\theta})$ be an optimal dual solution of (RLF2), corresponding to constraints (3.9), (3.10), (1.25), (1.26), (2.22), and (2.15) respectively. We say that a constraint is active if its value is non-zero in the dual solution. Let E be the set of active TLPC, and τ^e defines cut $e \in E$ with dual value $\bar{\zeta}_e$. Let N be the set of active RCC, and C^n defines cut $n \in N$ with dual value $\bar{\rho}_n$. Let M be the set of active R1C, and α^m defines cut $m \in M$ with dual value $\bar{\xi}_m$. Let H be the set of active SSI, and (S^h, C^h, β^h) defines cut $h \in H$ with dual value $\bar{\eta}_h$, where β^h is the coefficient of variable ν_S in this SSI.

First level pricing problem

The reduced cost of a first-level route $p \in P$ is equal to

$$\sum_{a \in A_1} f_a \tilde{x}_a^p - \sum_{e \in E:\, p \in P(\tau^e)} Q_1 \bar{\zeta}_e + \sum_{h \in H:\, S_p \cap S^h \neq \emptyset} \beta^h \bar{\eta}_h + \sum_{s \in S_p} \sum_{c \in \mathcal{C}} \bar{\theta}_{sc}. \tag{3.16}$$

We cannot express reduced cost (3.16) as a linear combination of reduced costs on arcs in A_1. Thus, solving the first-level pricing problem as a standard resource constrained shortest path problem (RCSPP) is not possible. Here, we take advantage of the fact that the number of depots and satellites in the literature instances is not large. We enumerate all feasible first-level routes before starting the column and cut generation. However, we cannot include all corresponding first-level route variables λ to (LF2), as their number can exceed 100,000. Instead, as proposed by Contardo and Martinelli (2014), we solve the first-level pricing problem by inspection of enumerated routes. The reduced cost of every enumerated route is updated based on the current dual solution, and routes with the smallest reduced costs are selected.

Second-level single-trip pricing problem

This problem can be decomposed in $|\mathcal{S}|$ independent subproblems, one per satellite. Given a satellite $s \in \mathcal{S}$, the reduced cost of a second-level single-trip route $r \in R_s$ is

calculated as

$$\sum_{a\in A_2} f_a\tilde{y}_a^r - \sum_{c\in\mathcal{C}}\sum_{a\in\delta_2^-(\{c\})}\bar{\pi}_c\tilde{y}_a^r - \sum_{n\in N}\sum_{a\in\delta_2^-(C^n)}\bar{\rho}_n\tilde{y}_a^r + \sum_{c\in\mathcal{C}}\sum_{a\in\delta_2^-(\{c\})}\bar{\theta}_{sc}\tilde{y}_a^r$$
$$- \sum_{h\in H:\, s\notin S^h}\sum_{a\in\delta_2^-(C^h)}\bar{\eta}_h\tilde{y}_a^r + \sum_{e\in E:\, r\in R(\tau^e)}\bar{\zeta}_e\tilde{d}^r + \sum_{m\in M}\left\lfloor\sum_{c\in\mathcal{C}}\alpha_c^m\tilde{z}_c^r\right\rfloor\bar{\xi}_m. \tag{3.17}$$

Consider first the case without active R1Cs, i.e. $M=\emptyset$. Reduced cost (3.17) cannot be expressed as a linear combination of reduced costs on arcs in A_2 because of the term coming from active TLPCs. Indeed, for each $e\in E$, the reduced cost of path $r\in R_s$ is increased by $\bar{\zeta}_e\tilde{d}^r$ if r departs from satellite s strictly before time moment τ_s^e. Therefore, we define a graph $\mathcal{G}^s$, which is extended from graph G_2, to express the contribution of TLPC to (3.17) as a linear combination of reduced costs on arcs in $\mathcal{G}^s$. Then, the second-level pricing subproblem corresponding to satellite s can be formulated as a standard RCSPP in extended graph $\mathcal{G}^s$.

Let $\bar{T}^s=(\bar{t}_0^s,\bar{t}_1^s,\ldots,\bar{t}_{\bar{n}(s)}^s)$, $\bar{t}_0^s=l_s+\sigma_s$, be the ordered set of different time moments τ_s^e for all $e\in E$, augmented by value $l_s+\sigma_s$ if necessary. All values τ_s^e which are less than $l_s+\sigma_s$ are ignored. Set of nodes in $\mathcal{G}^s$ is defined as $\mathcal{V}^s\cup\mathcal{V}^{\mathcal{C}}\cup\{v_{\text{source}},v_{\text{sink}}\}$, where node $v_k^s\in\mathcal{V}^s$, $0\le k\le\bar{n}(s)$, corresponds to the situation in which city freighter is available at time $\bar{t}_k^s$ at satellite s, and node $v_{cq}^{\mathcal{C}}\in\mathcal{V}^{\mathcal{C}}$, $c\in\mathcal{C}$, $d_c\le q\le Q_2$, corresponds to the situation in which vehicle is coming to customer c with load q. Set of arcs in $\mathcal{G}^s$ is defined as $\{(v_{\text{source}},v_0^s)\}\cup\mathcal{A}^s\cup\mathcal{A}^{s\to\mathcal{C}}\cup\mathcal{A}^{\mathcal{C}}\cup\mathcal{A}^{\text{sink}}$. Arcs in $\mathcal{A}^s=\{(v_{k-1}^s,v_k^s)\}_{1\le k\le\bar{n}(s)}$ connect consecutive nodes in $\mathcal{V}^s$. Arcs in $\mathcal{A}^{s\to\mathcal{C}}=\{(v,v')\}_{v\in\mathcal{V}^s,v'\in\mathcal{V}^{\mathcal{C}}}$ connect all satellite nodes to all customer nodes. Arcs in $\mathcal{A}^{\mathcal{C}}=\{(v_{c,q}^{\mathcal{C}},v_{c',q-d_c}^{\mathcal{C}})\}_{c,c'\in\mathcal{C},\,c\neq c',\,d_c+d_{c'}\le q\le Q_2}$ interconnect customer nodes. Finally, arcs in $\mathcal{A}^{\text{sink}}=\{(v_{c,d_c}^{\mathcal{C}},v_{\text{sink}})\}_{c\in\mathcal{C}}$ connect customer nodes to the sink. Each arc in $\mathcal{A}^{s\to\mathcal{C}}$ project into the corresponding arc in A_2 between satellite s and a customer. Each arc in $\mathcal{A}^{\mathcal{C}}$ projects into the corresponding arc in A_2 between two customers. Each arc in $\mathcal{A}^{\text{sink}}$ projects into the corresponding arc in A_2 between a customer and satellite s.

We now formulate the pricing problem as a RCSPP in graph $\mathcal{G}^s$. Time is the only resource. The time consumption of arc a in graph $\mathcal{G}^s$ is equal to the sum of travel time $t_{a'}$ and the service time of the satellite or customer corresponding to the tail of arc $a'\in A_2$ to which a projects. If a does not project to an arc in A_2, the time consumption is zero. Bounds on the accumulated time consumption are given on nodes. These bounds are $[0,0]$ for v_{source}, $[\bar{t}_k^s,u_s]$ for $v_k^s\in\mathcal{V}^s$, $[l_c,u_c]$ for nodes $v_{cq}^{\mathcal{C}}\in\mathcal{V}^{\mathcal{C}}$, and $[l_s+\sigma_s,u_s]$ for v_{sink}. The time resource is disposable, as defined by Pessoa et al. (2020): accumulated time consumption of a path in $\mathcal{G}^s$ at a node v is adjusted to the lower bound on the

accumulated time consumption at v, if the former is smaller than the latter. Figure 3.4 depicts an example of an extended graph $\mathcal{G}^s$. The reduced cost of each arc a in graph $\mathcal{G}^s$ is equal to the sum of the travelling cost $f_{a'}$ of arc $a' \in A_2$ to which a projects, the total coefficient of $\tilde{y}_a^r$ in (3.17), and the contribution of TLPCs. The reduced cost of an arc a is zero if a does not project to an arc in A_2. Contribution of active TLPCs to the reduced cost of each arc $(v_k^s, v_{c,q}^{\mathcal{C}}) \in \mathcal{A}^{s\to\mathcal{C}}$, $0 \le k \le \bar{n}(s)$, $c \in \mathcal{C}$, $d_c \le q \le Q_2$, is equal to $q \cdot \sum_{e\in E:\, \tau_s^e > \bar{t}_k} \bar{\zeta}_e$. Contribution of active TLPC to arcs which are not in $\mathcal{A}^{s\to\mathcal{C}}$ is zero.

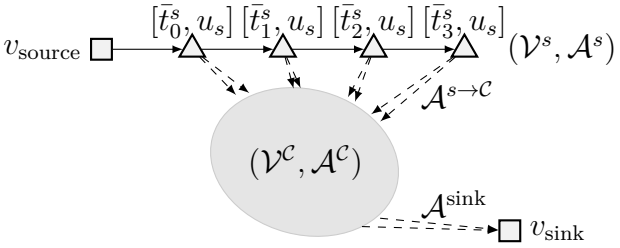


Figure 3.4: Example of extended graph $\mathcal{G}^s$ to price second-level single-trip routes

Second-level multi-trip pricing problem

This problem cannot be decomposed in subproblems. It is thus solved in one run. The reduced cost of a second-level multi-trip route $r \in R'$ is calculated as

$$\begin{aligned}\sum_{a\in A_2'} f_a \tilde{y}_a^r - \sum_{c\in\mathcal{C}} \sum_{a\in\delta_2^-(\{c\})} \bar{\pi}_c \tilde{y}_a^r - \sum_{n\in N} \sum_{a\in\delta_2^-(C^n)} \bar{\rho}_n \tilde{y}_a^r + \sum_{c\in\mathcal{C}} \sum_{s\in\mathcal{S}} \sum_{a\in\delta_2^-(\{c\})} \bar{\theta}_{sc} \tilde{y}_{sa}^r \\ - \sum_{h\in H} \sum_{s\in\mathcal{S}\setminus S^h} \sum_{a\in\delta_2^-(C^h)} \bar{\eta}_h \tilde{y}_{sa}^r + \sum_{e\in E} \sum_{i\in I_r(\tau^e)} \bar{\zeta}_e \tilde{d}^i + \sum_{m\in M} \left\lfloor \sum_{c\in\mathcal{C}} \alpha_c^m \tilde{z}_c^r \right\rfloor \bar{\xi}_m,\end{aligned} \tag{3.18}$$

If the problem is solved using graph representation (R1), values $\tilde{y}_{sa}^r$ are not available. Thus, SSIs and VSIs cannot be used and the corresponding terms in (3.18) are skipped.

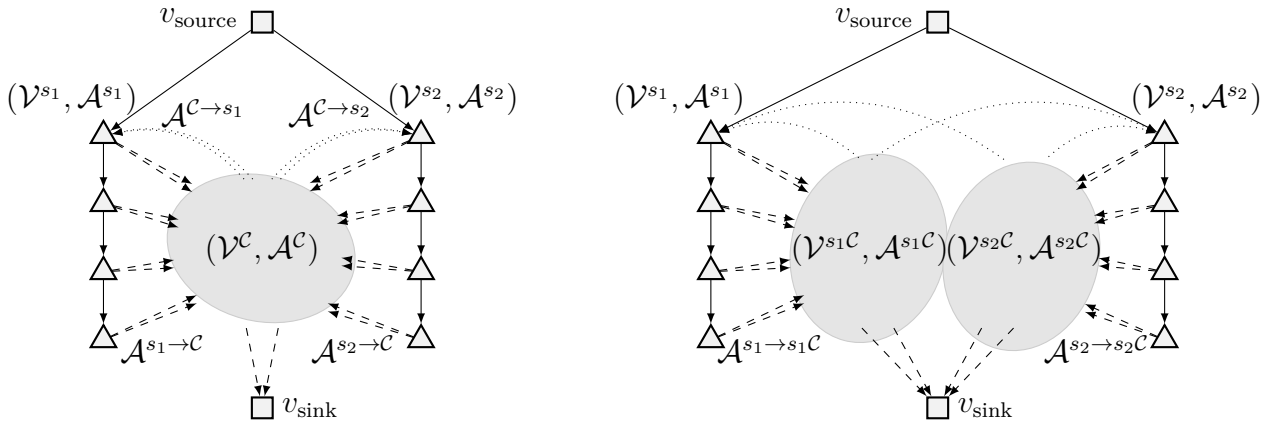
Similarly to the single-trip case and depending on the representation used, we extend graph G_2' or graph G_2'' to $\mathcal{G}'$ or $\mathcal{G}''$ respectively. We first describe graph $\mathcal{G}'$ extended from representation (R1). Set of nodes in $\mathcal{G}'$ is defined as $\bigcup_{s\in\mathcal{S}} \mathcal{V}^s \cup \mathcal{V}^{\mathcal{C}} \cup \{v_{\text{source}}, v_{\text{sink}}\}$, where $\mathcal{V}^s$, $s \in \mathcal{S}$, and $\mathcal{V}^{\mathcal{C}}$ are defined as in the single-trip case. Set of arcs in $\mathcal{G}'$ is defined as

$$\bigcup_{s\in\mathcal{S}} \left(\mathcal{A}^s \cup \mathcal{A}^{s\to\mathcal{C}} \cup \mathcal{A}^{\mathcal{C}\to s}\right) \cup \mathcal{A}^{\mathcal{C}} \cup \mathcal{A}^{\text{source}} \cup \mathcal{A}^{\text{sink}},$$

where $\mathcal{A}^s$, $\mathcal{A}^{s\to\mathcal{C}}$, $s \in \mathcal{S}$, $\mathcal{A}^{\mathcal{C}}$, and $\mathcal{A}^{\text{sink}}$ are defined as in the single-trip case. Given satellite $s \in \mathcal{S}$, arcs in $\mathcal{A}^{\mathcal{C}\to s} = \{(v_{c,d_c}^{\mathcal{C}}, v_0^s)\}_{c\in\mathcal{C}}$ connect some customer nodes to the initial

satellite s node, . Arcs in $\mathcal{A}^{\text{source}} = \{(v_{\text{source}}, v_0^s)\}_{s\in\mathcal{S}}$ connect the source to the initial satellite nodes. Projection of arcs in $\mathcal{A}^{s\to\mathcal{C}}$ and in $\mathcal{A}^{\mathcal{C}}$ is the same as in the single-trip case. Each arc in $\mathcal{A}^{\text{source}}$ projects into the corresponding arc in A_2' between the depot and a satellite. Each arc in $\mathcal{A}^{\text{sink}}$ projects into the corresponding arc in A_2' between a customer and the depot. Figure 3.5a depicts the structure of graph $\mathcal{G}'$.

The formulation of the RCSPP in graph $\mathcal{G}'$ is similar as the one in graph $\mathcal{G}^s$. Bounds on the accumulated time consumption are the same for nodes in $\bigcup_{s\in\mathcal{S}} \mathcal{V}^s \cup \mathcal{V}^{\mathcal{C}}$. Bounds for nodes $\{v_{\text{source}}, v_{\text{sink}}\}$ correspond to time window when the depot is open. The resource consumption of arc a in graph $\mathcal{G}'$ is equal to the sum of travel time $t_{a'}$ and the service time of the satellite or customer corresponding to the tail of arc $a' \in A_2'$ to which a projects. The reduced cost of each arc a in graph $\mathcal{G}'$ is equal to the sum of the travelling cost $f_{a'}$ of arc $a' \in A_2'$ to which a projects, the total coefficient of $\tilde{y}_a^r$ in (3.18), and the contribution of TLPCs. Contribution of active TLPCs to arcs in $\mathcal{A}^{s\to\mathcal{C}}$ is the same as in the single-trip case. Contribution of active TLPCs to other arcs is zero.



(a) Structure of graph $\mathcal{G}'$

(b) Structure of graph $\mathcal{G}''$

Figure 3.5: Examples of extended graphs to price second-level multi-trip routes

We now describe graph $\mathcal{G}''$ extended from representation (R2). The set of nodes in $\mathcal{G}''$ is the same as in $\mathcal{G}'$, except that customer nodes are duplicated for each satellite: $\bigcup_{s\in\mathcal{S}} \left(\mathcal{V}^s \cup \mathcal{V}^{s\mathcal{C}}\right) \cup \{v_{\text{source}}, v_{\text{sink}}\}$. Each node $v_{cq}^{s\mathcal{C}} \in \mathcal{V}^{s\mathcal{C}}$, $s \in \mathcal{S}$, $c \in \mathcal{C}$, $d_c \le q \le Q_2$, corresponds to the situation in which a city freighter is coming to customer c with load q, and the last visited satellite is s. Set of arcs in $\mathcal{G}''$ is defined as

$$\bigcup_{s\in\mathcal{S}} \left(\bigcup_{s'\in\mathcal{S}} \mathcal{A}^{s\mathcal{C}\to s'} \cup \mathcal{A}^s \cup \mathcal{A}^{s\to s\mathcal{C}} \cup \mathcal{A}^{s\mathcal{C}} \cup \mathcal{A}^{s\to\text{sink}} \right) \cup \mathcal{A}^{\text{source}},$$

where $\mathcal{A}^s$, $s \in \mathcal{S}$, and $\mathcal{A}^{\text{source}}$ are defined as for graph $\mathcal{G}'$. Given satellites $s, s' \in \mathcal{S}$, arcs in $\mathcal{A}^{s\mathcal{C}\to s'} = \{(v^{s\mathcal{C}}_{c,d_c}, v^{s'}_0)\}_{c\in\mathcal{C}}$ connect some customer nodes associated to satellite s to the initial satellite s' nodes. Arcs in $\mathcal{A}^{s\to s\mathcal{C}} = \{(v, v')\}_{v\in\mathcal{V}^s, v'\in\mathcal{V}^{s\mathcal{C}}}$, $s \in \mathcal{S}$, connect all nodes of satellite s to all customer nodes associated to s. Arcs in $\mathcal{A}^{s\mathcal{C}} = \{(v^{s\mathcal{C}}_{c,q}, v^{s\mathcal{C}}_{c',q-d_c})\}_{c,c'\in\mathcal{C},\, c\neq c',\, d_c+d_{c'}\leq q\leq Q_2}$, $s \in \mathcal{S}$, interconnect customer nodes associated to the same satellite s. Finally, arcs in $\mathcal{A}^{s\to\text{sink}} = \{(v^{s\mathcal{C}}_{c,d_c}, v_{\text{sink}})\}_{c\in\mathcal{C}}$, $s \in \mathcal{S}$, connect customer nodes associated to s to the sink. Projection of arcs in graph $\mathcal{G}''$ to arcs in A'_2 is similar to the projection of arcs in graph $\mathcal{G}'$. Figure 3.5b depicts the structure of graph $\mathcal{G}'$.

The formulation of the RCSPP in graph $\mathcal{G}'$ is similar to the one in graph $\mathcal{G}''$. Bounds on the accumulated time consumption are the same for nodes in $\bigcup_{s\in\mathcal{S}} \mathcal{V}^s \cup \{v_{\text{source}}, v_{\text{sink}}\}$. Bounds for each customer node in $\mathcal{V}^{s\mathcal{C}}$, $s \in \mathcal{S}$, are equal to the start and the end of time window of the corresponding customer. The time consumption of arc a in graph $\mathcal{G}''$ is equal to the sum of travel time $t_{a'}$ and the service time of the satellite or customer corresponding to the tail of arc $a' \in A'_2$ to which a projects. The reduced cost of each arc a in graph $\mathcal{G}''$ is equal to the sum of the travelling cost $f_{a'}$ of arc $a' \in A'_2$ to which a projects plus the total coefficient of $\tilde{y}^r_{a'}$ in (3.18), the contribution of TLPC , and the contribution of SSI and VSI. Contribution of active TLPC to arcs in $\mathcal{A}^{s\to s\mathcal{C}}$, $s \in \mathcal{S}$, is the same as in the single-trip case. Contribution of active TLPC to other arcs is zero. Contribution of active SSI and VSI to arc $a \in \mathcal{A}^{s\mathcal{C}}$, $s \in \mathcal{S}$, is equal to the total coefficient of $\tilde{y}^r_{a's}$, where a' the arc in A'_2 to which a projects.

3.3.2 TLPC separation algorithm

Given a solution to formulation (RLF2), the TLPC separation algorithm searches for violated TLPCs. These constraints are essential to the formulation. Thus, the separation algorithm should find a violated constraint when it exists. Our algorithm first finds the most violated constraint, and then it tries heuristically to obtain other violated constraints. We now present the algorithm for the single-trip case. Extension to the multi-trip case is obvious after replacing second-level routes with second-level trips.

Our separation algorithm is based on the proof of Proposition 5. Given fractional or integer solution $(\bar{\lambda}, \bar{\mu})$, we construct graph $\bar{G}$, as described in the proof. We then find a minimum cut in this graph. If the value of this cut is equal to $d(\mathcal{C})$, then no TLPC violated by $(\bar{\lambda}, \bar{\mu})$ exists, and the algorithm stops. If the value of the cut is strictly smaller than $d(\mathcal{C})$, we obtain set $\bar{P}'$ of first-level routes and set $\bar{R}'$ of second-level routes as defined in the proof. Vector $\bar{\tau}$ characterising the most violated constraint is then calculated according to formula (3.14).

If a violated TLPC is found, we try to obtain other violated constraints. For this, we define the directed graph $\vec{G} = (\vec{V}, \vec{A})$ that represents precedence relations between first-level routes. Remember that $\bar{P} = \{p \in P : \bar{\lambda}_p > 0\}$ and $\bar{R} = \{r \in R : \bar{\mu}_r > 0\}$ are the sets of first-level and second-level routes participating in the solution. Let also $\vec{T}^s = (\vec{t}^s_1, \vec{t}^s_2, \ldots, \vec{t}^s_{\vec{n}(s)})$ be the ordered set of different time moments at which first-level routes in $\bar{P}$ depart from satellite $s \in \mathcal{S}$. We have $\vec{V} = \vec{V}^P \cup \left(\cup_{s\in\mathcal{S}} \vec{V}^s\right)$, where node $\vec{v}^P_p \in \vec{V}^P$, $p \in \bar{P}$, corresponds to a first-level route participating in the solution, and node $\vec{v}^s_k$, $1 \le k \le \vec{n}(s)$, corresponds to a visit of a first-level route in the solution to satellite s. Set of arcs $\vec{A}$ is defined as $\left(\cup_{p\in\bar{P}} \vec{A}^p\right) \cup \left(\cup_{s\in\mathcal{S}} \vec{A}^s\right)$. Subset $\vec{A}^p$ of arcs connects node $\vec{v}^P_p$ with the corresponding visits or route p to satellites: $\vec{A}^p = \left\{(\vec{v}^P_p, \vec{v}^s_{k(p,s)}), (\vec{v}^s_{k(p,s)}, \vec{v}^P_p)\right\}_{s\in S_p}$, where $k(p,s)$ is the index in $\vec{T}^s$ of the time moment when route p departs from satellite s. Subset $\vec{A}^s$ of arcs connects consecutive nodes corresponding to visits of routes to satellite s in the reverse chronological order: $\vec{A}^s = \{\vec{v}^s_{k+1}, \vec{v}^s_k\}_{1\le k<\vec{n}(s)}$. As an example, graph $\vec{G}$ corresponding to Example 6 is depicted in Figure 3.6.

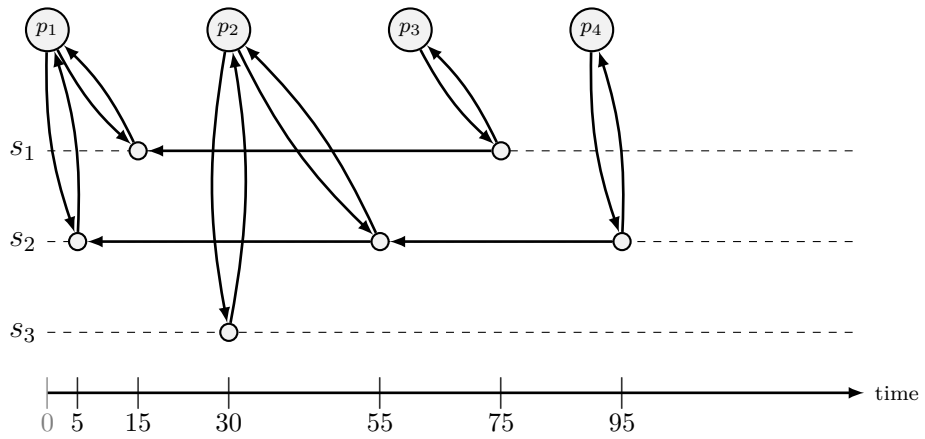


Figure 3.6: Graph $\vec{G}$ corresponding to Example 6

Using graph $\vec{G}$, for each first-level route $p \in \bar{P}'$, we find set $\vec{P}^p$ of first-level routes in $\bar{P}$ which "precede" p. Set $\vec{P}^p$ corresponds to all nodes in $\vec{v}^P_p$ which are reachable from node $\vec{v}^P_p$. In the example in Figure 3.6, we have $\vec{P}^{p_1} = \{p_1\}$, $\vec{P}^{p_2} = \{p_1, p_2\}$, $\vec{P}^{p_3} = \{p_1, p_3\}$, and $\vec{P}^{p_4} = \{p_1, p_2, p_4\}$. For each set $\vec{P}^p$, $p \in \bar{P}'$, we then find the vector τ^p such that $\bar{P} \cap P(\tau^p) = \vec{P}^p$ and set $\bar{R} \cap R(\tau^p)$ is as large as possible so that the violation of the corresponding TLPC is maximized. Component τ^p_s, $s \in \mathcal{S}$, of such vector τ^p, $p \in \bar{P}'$, is calculated as $\tau^p_s = \min\left\{u_s, \min_{p\in\bar{P}\setminus\bar{P}':\, s\in S_p}\{\vec{t}^s_{k(p,s)}\}\right\}$. For each $p \in \bar{P}'$ we verify whether constraint (3.10) characterized by τ^p is violated. All violated TLPCs are then added to formulation (RLF2).

3.3.3 The overall algorithm

The structure of the overall branch-cut-and-price (BCP) algorithm we use is similar to the one presented in Section 1.3.4 with the parameters presented in Section 2.4. Thus, we give more details only on branching strategies, which are problem-specific.

Suppose that the solution $(\bar{\lambda}, \bar{\mu})$ obtained by column and cut generation at a node of the branch-and-bound tree is fractional. As described in Section 3.2.3, values $\bar{x}$, $\bar{y}$, and $\bar{\nu}$ are computed based on $\bar{\lambda}$ and $\bar{\mu}$. We branch on (i) the number of urban trucks $\sum_{p \in P} \bar{\lambda}_p$; (ii) the number of city freighters $\sum_{r \in R} \bar{\mu}_r$ or $\sum_{r \in R'} \bar{\mu}_r$; (iii) the number of urban trucks visiting a subset of satellites ν_S, $S \subseteq \mathcal{S}$; (iv) the use of first-level arcs $\bar{x}_a$, $a \in A_1$, by urban trucks; and (v) the use of second-level arcs $\bar{y}_a$, $a \in A_2$ or $a \in A_2'$, by city freighters. In the single-trip case, we also branch on the number of city freighters starting from a satellite $\sum_{r \in R:\, \tilde{s}_r = s} \bar{\mu}_r$, $s \in \mathcal{S}$. The multi-phase strong branching procedure, described in Sadykov et al. (2020), selects the branching candidate.

3.3.4 Post-processing

The post-processing phase seeks to synchronize the arrival of urban trucks and city freighters at satellites, if possible. In other words, given an optimal solution (λ^*, μ^*) to (F2), it modifies the arrival and departure times of routes such that the storage usage while transferring the freight from urban trucks to city freighters is minimized. For the sake of brevity, we focus on the multi-trip variant. Adjustments for the single-trip variant are straightforward.

Let $P^* = \{p \in P : \lambda_p^* \geq 1\}$ and $R^* = \{r \in R' : \mu_r^* = 1\}$. Let also P_i^* be the set of routes in P^* which can serve trip $i \in I_r$, $r \in R^*$: $P_i^* = \{p \in P^* : \exists k, 1 \leq k < n(p),\ v_k^p = \tilde{s}^i,\ \tilde{t}_k^p \leq \tilde{t}^i\}$. We denote as $k^1(p,i)$ the index number of visit to satellite $\tilde{s}^i$, $i \in I_r$, $r \in R^*$, in route $p \in P_i^*$. We also denote as $k^2(r,i)$ the index number of visit to satellite $\tilde{s}^i$, $i \in I_r$, in route $r \in R^*$ when starting trip i.

Let binary variable χ_{pji}, $i \in I_r$, $r \in R^*$, $p \in P_i^*$, $1 \leq j \leq \lambda_p^*$, be equal to one if trip i is served by the j-th vehicle following first-level route p. Let variable γ_{pji}, $i \in I_r$, $r \in R^*$, $p \in P_i^*$, $1 \leq j \leq \lambda_p^*$, be equal to the fraction of the load of trip i served by the j-th vehicle following first-level route p. Let variable Δ_{pji}, $i \in I_r$, $r \in R^*$, $p \in P_i^*$, $1 \leq j \leq \lambda_p^*$, be equal to the time elapsed between the departure of the j-th vehicle on route p and the departure of trip i in satellite $\tilde{s}^i$, if trip i is served by this vehicle. If route r leaves satellite $\tilde{s}^i$ before departure of the j-th vehicle on route p or trip i is not served by this vehicle, then $\Delta_{pji} = 0$. Let variables ϕ_{pjk}^{1-} and ϕ_{pjk}^{1+} be equal to the arrival and departure times of the j-th first-level vehicle on route $p \in P^*$ at node $v_k^p \in \{0\} \cup \mathcal{S}$, $0 \leq k \leq n(p)$.

Let variable ϕ^{2+}_{rk} be the departure time of second-level route $r \in R^*$ at node $v^r_k \in \mathcal{S} \cup \mathcal{C}$, $0 \leq k \leq n(r)$. The following mixed integer linear program minimizes the total time during which satellites store freight.

$$(PP) \equiv \min \quad \sum_{r \in R^*} \sum_{i \in I_r} \sum_{p \in P^*_i} \sum_{j=1}^{\lambda^*_p} \Delta_{pji} \tag{3.19}$$

$$\text{s.t.} \quad \gamma_{pji} \leq \chi_{pji} \qquad r \in R^*,\ i \in I_r,\ p \in P^*_i,\ 1 \leq j \leq \lambda^*_p \tag{3.20}$$

$$\sum_{r \in R^*} \sum_{i \in I_r:\ p \in P^*_i} \tilde{d}^i \gamma_{pji} \leq Q_1 \qquad p \in P^*,\ 1 \leq j \leq \lambda^*_p \tag{3.21}$$

$$\sum_{p \in P^*_i} \sum_{j=1}^{\lambda^*_p} \gamma_{pji} = 1 \qquad r \in R^*,\ i \in I_r \tag{3.22}$$

$$\Delta_{pji} - \phi^{2+}_{r,k^2(r,i)} + \phi^{1+}_{p,j,k^1(p,i)} + (u_{\tilde{s}^i} - l_{\tilde{s}^i}) \cdot (1 - \chi_{pji}) \geq 0 \qquad r \in R^*,\ i \in I_r,\ p \in P^*_i,\ 1 \leq j \leq \lambda^*_p \tag{3.23}$$

$$\phi^{2+}_{r,k^2(r,i)} - \phi^{1-}_{p,j,k^1(p,i)} - \sigma_{\tilde{s}^i} + (u_{\tilde{s}^i} - l_{\tilde{s}^i}) \cdot (1 - \chi_{pji}) \geq 0 \qquad r \in R^*,\ i \in I_r,\ p \in P^*_i,\ 1 \leq j \leq \lambda^*_p \tag{3.24}$$

$$\phi^{1+}_{p,j,k-1} + t_{(v^p_{k-1}, v^p_k)} \leq \phi^{1-}_{p,j,k} \qquad p \in P^*,\ 1 \leq k \leq n(p),\ 1 \leq j \leq \lambda^*_p \tag{3.25}$$

$$\phi^{1-}_{p,j,k} + \sigma_{v^p_k} \leq \phi^{1+}_{p,j,k} \qquad p \in P^*,\ 1 \leq k < n(p),\ 1 \leq j \leq \lambda^*_p \tag{3.26}$$

$$\phi^{2+}_{r,k-1} + t_{(v^r_{k-1}, v^r_k)} + \sigma_{v^r_k} \leq \phi^{2+}_{r,k} \qquad r \in R^*,\ 1 \leq k \leq n(r) \tag{3.27}$$

$$l_{v^p_k} \leq \phi^{1-}_{p,j,k} \leq u_{v^p_k} \qquad p \in P^*,\ 0 \leq k \leq n(p),\ 1 \leq j \leq \lambda^*_p \tag{3.28}$$

$$l_{v^r_k} \leq \phi^{2+}_{p,k} - \sigma_{v^r_k} \leq u_{v^r_k} \qquad r \in R^*,\ 0 \leq k \leq n(r) \tag{3.29}$$

$$\chi_{pji} \in \{0, 1\} \qquad r \in R^*,\ i \in I_r,\ p \in P^*_i,\ 1 \leq j \leq \lambda^*_p \tag{3.30}$$

$$0 \leq \gamma_{pji} \leq 1 \qquad r \in R^*,\ i \in I_r,\ p \in P^*_i,\ 1 \leq j \leq \lambda^*_p \tag{3.31}$$

$$\Delta_{pji} \geq 0 \qquad r \in R^*,\ i \in I_r,\ p \in P_i^*,\ 1 \leq j \leq \lambda_p^* \tag{3.32}$$

The objective function (3.19) minimises the total number of time units during which freight is stored at satellites. Constraints (3.20) link variables χ and γ. Constraints (3.21) ensure that the capacity of the first-level vehicles is satisfied. Constraints (3.22) ensures that each trip of city freighters receives the desired amount of freight. Constraints (3.23) compute the values of variables Δ: if a trip i is served by the j-th first-level vehicle on path p, then Δ_{pji} is not smaller than the difference between departure times of trip i and the vehicle from satellite $\tilde{s}^i$. In these and in the next constraints, expression $(u_{\tilde{s}^i} - l_{\tilde{s}^i})$ acts as a big-M value. Constraints (3.24) ensures that each first-level route p arrives and completes its service time before all departures of the second-level trip it serves. Constraints (3.25)–(3.27) guarantee that arrival and departure times of first-level and second-level vehicles are compatible with the visited order of nodes. Constraints (3.28) and (3.29) ensure that all time windows are satisfied. If the optimal solution value is zero, then we can synchronize urban trucks and city freighters, and solution (λ^*, μ^*) is feasible and optimal for the case in which satellites do not have storage. Otherwise, the value of solution (λ^*, μ^*) provides a lower bound for the case without storage.

If storage is not needed, then we can further check if freight consolidation can be avoided, i.e. if each second-level trip can be served by only one first-level vehicle. To do it, we should verify if there exists a feasible solution to formulation (3.21)–(3.30), in which variables Δ are fixed to 0 in constraints (3.23), and variables γ_{pji} are replaced by variables χ_{pji} in constraints (3.21)–(3.22).

3.4 Computational results

The implementation of the proposed algorithm was done in C++ language. We used the following third-party libraries and codes:

- BaPCod C++ library (Vanderbeck et al., 2019) which implements the BCP framework;
- C++ code, developed by Sadykov et al. (2020), which implements the bucket graph based labeling algorithm, bucket arc elimination procedure, elementary route enumeration, and the separation algorithm for R1Cs;
- CVRPSEP C++ library (Lysgaard, 2018) which implements heuristic separation of RCCs;

- IBM CPLEX Optimizer version 12.10 as the LP solver in column generation, as the solver for the enumerated MIPs, and as the solver for the MIP post-processing.

Experiments were run on a 2 Dodeca-core Haswell Intel Xeon E5-2680 v3 servers at 2.5 GHz. Every server has 128 Go of RAM. Each instance is solved on a single thread.

3.4.1 Literature Instances

In this chapter, we experiment our algorithm on single-trip instances proposed by Dellaert et al. (2019) and multi-trip instances proposed by Grangier et al. (2016). Three main characteristics allow us to estimate the difficulty of an instance: the size (number of customers, satellites, and depots), the capacity of city freighters, and the size of time windows relative to the time horizon. Indeed, a large second-level vehicle capacity results in large extended graphs $\mathcal{G}^s$, $\mathcal{G}'$, and $\mathcal{G}''$, used in the pricing problems. Also, wide time windows lead to a larger number of labels in the labelling algorithm.

Instances by Dellaert et al. (2019) are divided into four classes Ca, Cb, Cc, and Cd. They have narrow time windows: the widest time window is from 7% to 20% of the time horizon, depending on the instance class. Instances in classes Ca, Cb, and Cd have customers with demands 10 or 20 with city freighter capacity equal to 50. Thus, capacity can be divided by 10. Instances in class Cc has customers with integer demands from 5 to 25, and city freighter capacity equal to 50. We put all instances by Dellaert et al. (2019) in set D.

Instances by Grangier et al. (2016) were adapted from famous Solomon instances for the VRPTW. We split these instances into three sets depending on their difficulty. Set G contains 9 difficult instances in class `c1`, `c102`, ..., `c109`, which have small city freighter capacity (the original capacity can also be divided by 10) and tight time windows. Set H contains very difficult instances in classes `c2`, `r1`, and `rc1`, which have either wide time windows or large city freighter capacity. Set I contains "intractable" instances in classes `r2` and `rc2` with wide time windows and large city freighter capacity.

Table 3.1: Sets of instances from the literature used for experiments

Set	#	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{C}\rvert$	Difficulty	Authors
D	237	2,3,6	3,4,5	15,30,50,100	easy-difficult	Dellaert et al. (2019)
G	9	1	8	100	difficult	Grangier et al. (2016)
H	28	1	8	100	very difficult	Grangier et al. (2016)
I	19	1	8	100	intractable	Grangier et al. (2016)

Table 3.1 gives an overview of these instances. It contains, for each set, the number of instances, the number of distribution centers, the number of satellites, the number of

customers, difficulty estimation, and the authors. The number of feasible first-level routes starting from a distribution center is bounded from above by $\sum_{k=1}^{|\mathcal{S}|} \binom{|\mathcal{S}|}{k} k!$, where $|\mathcal{S}|$ is the number of satellites. Thus instances in set D have at most 1950 feasible first-level routes. Instances in other sets G, H, and I may have up to 109,600 feasible first-level routes. It is important to notice that in all literature instances, the capacity of an urban truck is a multiple of the capacity of a city freighter.

3.4.2 Results for literature instances

We first experiment our algorithm on literature instances. We use two variants of our BCP algorithm :

- BCP_{base} — the variant without separation of valid inequalities SSI and VCI (thus, smaller graph $\mathcal{G}'$ is used when solving pricing problems for multi-trip instances)
- $\text{BCP}_{\text{complete}}$ — the variant with separation of all families of valid inequalities

Results for single-trip instances

We run our BCP algorithm on instances of set D with the time limit of 10 hours. On each server, we optimize in parallel 12 instances that share 128 Go of RAM. Table 3.2 compares two variants of our algorithm with the one proposed by Dellaert et al. (2019). For a fair comparison, the solution time of the latter is multiplied by 1.2 because of the difference in speed between the computers used. In the table, we give average values for instances with the same number of distribution centers, satellites, and customers: the average root gap (RG), the geometric mean of the number of branch-and-bound nodes (Nds), the geometric mean of total solution time in seconds (ST), and the number of instances solved to optimality within 3 hours. For unsolved instances, the solution time is set to 3 hours.

We first discuss the comparison between two variants of our BCP algorithm. It is clear that the separation of SSIs and VCIs makes the algorithm more efficient. Indeed, these cuts improve dramatically the root gap and significantly decrease the number of nodes in the branch-and-bound tree. Four more instances could be solved to optimality in 3 hours, and the average solution time is several times smaller. The complete variant of our BCP algorithm solves all but four instances within three hours. Two additional instances are solved in 10 hours, and two instances remain open.

Both variants of our BCP algorithm outperform significantly the algorithm by Dellaert et al. (2019). Even though we solve a relaxation of the problem solved by Dellaert

Table 3.2: Comparison with the algorithm by Dellaert et al. (2019)

Instance			BCP_{base}				$BCP_{complete}$				Literature	
$\|\mathcal{D}\|$	$\|\mathcal{S}\|$	$\|\mathcal{C}\|$	RG(%)	Nds	ST(s)	Solved	RG(%)	Nds	ST(s)	Solved	ST(s)	Solved
2	3	15	0.49	1.1	1	20/20	0.00	1.0	1	20/20	0	20/20
2	3	30	2.31	3.9	6	20/20	0.12	1.7	3	20/20	14	20/20
2	3	50	0.87	4.1	17	20/20	0.27	2.5	14	20/20	715	14/20
2	3	100	0.46	51.0	532	16/20	0.33	13.1	195	18/20	7780	6/20
3	5	15	3.10	1.7	6	20/20	0.05	1.1	4	20/20	2	20/20
3	5	30	3.93	6.5	35	20/20	0.23	1.7	13	20/20	50	20/20
3	5	50	2.34	22.8	232	20/20	0.37	2.8	51	20/20	862	19/20
3	5	100	0.66	42.4	986	17/20	0.34	12.2	449	19/20	10152	3/20
6	4	15	1.19	1.1	3	17/17	0.00	1.0	3	17/17	0	17/17
6	4	30	3.15	4.4	21	20/20	0.17	1.6	9	20/20	17	20/20
6	4	50	0.89	5.3	49	20/20	0.30	2.9	33	20/20	586	18/20
6	4	100	0.39	12.4	283	19/20	0.27	7.2	215	19/20	10715	2/20

et al. (2019), our post-processing procedure shows that all our optimal solutions can be transformed to satisfy the exact synchronization and avoid freight consolidation without increasing the transportation cost. Thus, all our optimal solutions are also optimal for the variant of the problem considered by Dellaert et al. (2019). We solve 54 open instances to optimality for the first time. The best solutions we found for two instances not solved to optimality require freight consolidation. Thus, these solutions are not feasible for the variant considered by Dellaert et al. (2019).

Multi-trip variant

We run our BCP algorithm on multi-trip instances with the time limit of 10 hours. On each server, we optimize 2 instances of set G that share 128 Go using $BCP_{complete}$, and only one instance of set H using BCP_{base}. Variant $BCP_{complete}$ is not suitable for instances in set H, because graph $\mathcal{G}''$ in the pricing problem becomes very large, and the pricing problem becomes intractable. The same happens for instances in set I: even graph $\mathcal{G}'$ used in variant BCP_{base} becomes too large. We fix the sizes of the fleets of urban trucks and city freighters to sizes in the best solutions found by Grangier et al. (2016).

Table 3.3 gives an overview of our results. For each set, it contains the number of instances solved to optimality, the number of instances for which the algorithm finds a feasible solution without proving optimality, the number of instances for which the algorithm does not find any solution, and the number of instances on which the algorithm fails, i.e. the column generation does not finish when solving formulation (LF2) and a lower bound cannot be obtained.

Table 3.3: Overview of results for 2E-VRPTW multi-trip instances

Set	Algorithm	Optimal	Feasible	No solution	Failure	Total
G	$BCP_{complete}$	5	2	2	0	9
H	BCP_{base}	3	5	17	3	28
I	BCP_{base}	0	0	0	19	19

Since our algorithm finds optimal solutions to half of the instances of set G, it can handle multi-trip instances with small city freighter capacity and tight time windows. Other instances are much more difficult for our algorithm. Indeed, BCP_{base} finds only 3 optimal solutions for instances in set H and fails to optimize the root node for 3 of them. For the instances in set I, the model does not fit in the server memory.

Table 3.4: Overview of experiments on multi-trip instances

Instance	Set	Δ	Consolidation	BCP val	Grangier et al. (2016)	Gap(%)
c101	G	1478	false	1969.3	2022.4	2.70
c105	G	662	false	1873.3	1934.0	3.24
c106	G	997	false	1903.0	1945.0	2.21
c107	G	557	false	1846.4	1888.9	2.30
c108	G	756	false	1825.9	1875.3	2.71
c201	H	5763	false	1277.5	1389.3	8.75
r101	H	0	false	**2298.7**	2333.5	1.51
r102	H	0	false	**2109.3**	2136.8	1.30

Table 3.4 lists the multi-trip instances solved to optimality by our BCP algorithm. In this table, we give the name of the instance, the set to which belongs the instance, the objective value of the post-processing MIP, the necessity of consolidation, the optimal value found by our algorithm, the best solution value found by Grangier et al. (2016), and the relative gap between these two values. Only for instances `r101` and `r102`, the optimal solutions do not require any storage. Thus, these solutions are also optimal for the variant, considered by Grangier et al. (2016).

Table 3.4 shows that the heuristic from Grangier et al. (2016) seems to be of a good quality. Indeed, the total distance travelled in the optimal solutions after the relaxation of exact synchronization is generally 2–3% lower than the one in the heuristic solutions. However, further progress in the exact solution of the 2E-VRPTW is needed to be able to estimate the quality of this heuristic on a larger set of instances. We also note that for instance `c201`, the gap between solutions with and without synchronisation is sufficiently large to consider the possibility to have storage at satellites, i.e. to replace satellites with UCCs in practice.

3.4.3 Results for new single-trip instances

As almost all single-trip literature instances were solved to optimality, here we generate more difficult instances. These instances are based on the multi-trip instances in sets G and H. The fleet size is unlimited, but the cost of using an urban truck is set to 50 and the cost of using a city freighter is set to 25, as in the instances in set D. The city freighters start and finish from a satellite, as in set D instances.

We run our algorithm with a time limit of 10 hours. On one server, two instances in set G on one instance in set H are optimized in parallel.

Table 3.5: Overview of experiments on new 2E-VRPTW single-trip instances

	Set G		Set H		
	Multi-trip	Single-trip	Multi-trip	Single-trip	
Status	$BCP_{complete}$	$BCP_{complete}$	BCP_{base}	BCP_{base}	$BCP_{complete}$
Optimal	5	9	3	10	15
Feasible	2	0	5	17	10
No solution	2	0	17	1	3
Failure	0	0	3	0	0

In Table 3.5, we report the overview of results for new single-trip instances. For comparison purposes, results for multi-trip instances are also recalled. This experiment shows that the multi-trip variant is more difficult than the single-trip one. Moreover, new single-trip instances are more difficult that instances in set D, as our algorithm solved to optimality only half of the instances in set H.

3.4.4 Results for smaller multi-trip instances

We derive new multi-trip instances from ones in sets G, H, and I, originally based on Solomon instances for the VRPTW. Positions of the distribution center and the satellites follow the procedure described by Grangier et al. (2016) that we recall now. They introduce an X/Y/M/N notation, where X and Y give the position of the distribution center expressed as a percentage of the size map, M and N are the number of rows and columns of a grid cutting the map in rectangles of equal sizes. Satellites are positioned at each intersection in the grid. In our new instances, we keep the distribution center in the same location but we change the number of customers and the number of satellites. We have :

- 25 customers with a 50/150/2/2 configuration (4 satellites)
- 50 customers with a 50/150/2/2 configuration (4 satellites)

- 75 customers with a 50/150/2/3 configuration (6 satellites)

The size of the vehicle fleet is unlimited. We set the cost of using an urban truck to 50 and the cost of using a city freighter to 25.

We run the variant $\text{BCP}_{\text{complete}}$ of our algorithm with the time limit of 10 hours. On each server, we simultaneously optimize at most 24 instances with 25 customers on a server, 12 instances with 50 customers, and 4 instances with 75 customers.

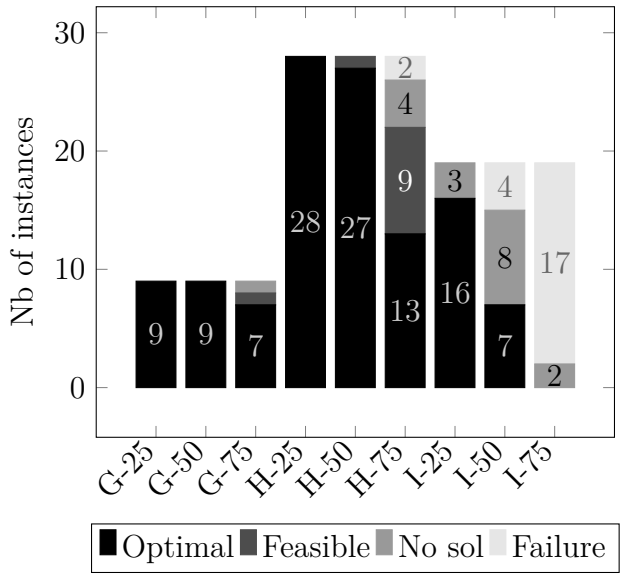


Figure 3.7: Overview of the results for multi-trip instances with 25, 50, and 75 customers

Figure 3.7 presents an overview of the results. The columns give the results for different instance classes denoted by the instance set and the number of customers. Our algorithm can solve the absolute majority of multi-trip instances with up to 50 customers. Beyond that size, the efficiency of the algorithm degrades significantly. Unsurprisingly, instances in set I become quickly intractable, even lower bounds for some instances in this set with 50 customers cannot be found.

3.4.5 Results for instances with modified vehicle capacity

In all instances considered above, the capacity of an urban truck is a multiple of the city freighter capacity. In this special case, freight consolidation at satellites is not likely to happen. Our experiments confirm this, as only for two instances freight consolidation is required in the best-found solutions.

In this experiment, we verify whether the change of vehicle capacity increases freight consolidation. We consider modified instances based on ones in sets D, G-75, and H-75.

For instances in set D, the capacity of urban trucks is reduced from 200 to 180, and the capacity of city freighters remains 50. Thus an urban truck has 3.6 times more capacity than a city freighter. For instances in sets G and H, we change the ratios for urban truck capacity / city freighter capacity, defined by Grangier et al. (2016). The ratio become 3.75/0.5 for instances in classes `r1`, `c1`, and `rc1` (i.e. an urban truck has 7.5 times more capacity than a city freighter). The ratio becomes 2/0.35 for instances in classes `r2`, `c2`, and `rc2` (i.e. an urban truck has 5.7 times more capacity than a city freighter). These new instances are run in the same way as the original instances. We report an overview of results in Tables 3.6 and 3.7.

Table 3.6: Overview of results for 2E-VRPTW instances with original vehicle capacity

Variant	Synchronization	No consolidation	Total Optimal
Single-trip	235	235	235
Multi-Trip	2	18	20

Table 3.7: Overview of results for 2E-VRPTW instances with modified vehicle capacity

Variant	Synchronization	No consolidation	Total Optimal
Single-trip	217	149	217
Multi-Trip	3	16	20

The first result is that about 10% of single trip instances are not solved to optimality. Thus, new instances are more difficult. Moreover, more than 25% of optimal single-trip solutions involve freight consolidation. This consolidation happens mostly for large instances with 100 customers. For multi-trip instances, it is difficult to draw any conclusions. Consolidation is required for 4 optimal solutions instead of 2, but this consolidation increase is small.

Chapter 4

Location-Routing and Related Problems

Location-routing problems (LRPs) arise when combining two classic combinatorial optimization problems: facility location and vehicle routing. In fact, the integration of both levels of decisions, depot location and vehicle routing, makes the LRP an interesting model for several practical applications, from the design of telecommunications networks to the operation of very competitive supply chains. Making decisions on the location of depots and the routing of vehicles independently usually leads to strongly suboptimal planning results, as observed by Salhi and Rand (1989). As a result, LRPs have been extensively studied in the literature as the latest survey paper Schneider and Drexl (2017) describes. The importance of LRPs is currently rising due to the surge of home delivery services and e-commerce. In those contexts, solving LRPs help in determining the location of urban depots from which customers would be served on vehicle routes.

This chapter mainly focuses on the *Capacitated Location-Routing Problem* (LRP for short). It consists of determining the depots opened together with the optimal routes starting and finishing at an opened depot, and visiting customers such that vehicle capacities are not exceeded, customer demands are satisfied, and the number of goods leaving a depot does not exceed the capacity of the depot.

The LRP is important not only from the practical point of view but also from the theoretical one. The LRP has the following *nested knapsack* structure. The set of customers are first assigned to feasible vehicle routes that have knapsack-type vehicle capacity constraints. Then the routes themselves are assigned to depots subject to knapsack-type depot capacity constraints. Such a nested knapsack structure is encountered in several problems in addition to the LRP and its variants. We mention some of these problems in the literature review in Section 4.1.

In this chapter, we propose a branch-cut-and-price algorithm for the LRP and some related problems with the nested knapsack structure. The algorithm is based on an integer programming formulation of the problem, in which each binary variable represents a feasible route having the lower-level knapsack structure. The formulation has set-partitioning and higher-level knapsack constraints. We show how to adapt some valid inequalities encountered in the literature. We also present two new families of valid inequalities. One of these families contains *Route Load Knapsack Cuts* (RLKCs) which can be used to solve many problems with the nested knapsack structure using a branch-cut-and-price algorithm. Our algorithm solves to optimality, for the first time, some of the classic LRP instances which remained open since 1999 and 2006, and also improves the best-known solutions for many other instances with up to 300 customers and 20 depots. Our experiments show that new families of valid inequalities make the algorithm more efficient. We also apply our algorithm to two related routing problems with the nested knapsack structure. The first problem is the capacitated vehicle routing problem with multiple capacitated depots (CVRP-CMD), encountered as a subproblem when solving the two-echelon stochastic multi-period capacitated location-routing problem by a logic-based Benders decomposition algorithm (Ben Mohamed et al., 2019). The second problem is the capacitated vehicle routing problem with time windows and shifts (VRPTW-S) proposed by Dabia et al. (2019). In the latter, the routes are assigned to shifts. This problem has an additional knapsack-type constraint on the total amount of freight loaded to vehicles during a shift. We computationally show that the family of RLKCs contributes to the efficiency of our algorithm applied to these problems.

We now describe the structure of the chapter. Section 4.1 reviews the literature on the LRP and related problems with the nested knapsack structure. Section 4.2 gives a route-based integer programming formulation for the LRP. We present how to adapt the formulation to the CVRP-CMD and the VRPTW-S. We also introduce new families of valid inequalities for the LRP and describe how to separate them. Section 4.3 presents how to solve the pricing problem for column generation. Section 4.4 presents the results of computational experiments for all three problems.

4.1 Literature review

The idea of combining together both levels of decision, depot location and vehicle routing, is not new. The first exact method for the LRP is due to Laporte and Nobert (1981) in which the authors develop a Branch-and-Cut algorithm to solve a special case of LRP where a single depot must be opened among a list of possible depot locations. Afterward,

Laporte et al. (1986) investigate the LRP with multi-depots to be opened and subject to vehicle capacities. The computational results reported show that they were able to solve instances with 8 depot locations and 20 customers. In a later work, Laporte et al. (1988) discuss the LRP in a context of asymmetrical costs, where vehicle capacities are replaced by constraints on the maximum length of the routes. Instances are then solved by a Branch-and-Cut algorithm.

Belenguer et al. (2011) use a two-index formulation for the LRP. By adapting some of the valid inequalities from the CVRP literature, together with others conceived specifically for the LRP formulation, the authors devise a Branch-and-Cut algorithm. Their approach can solve instances with 5 facilities and 50 customers. Contardo et al. (2013a) extend the work of Belenguer et al. (2011) and present four different arc-flow formulations for which they derive several new families of valid inequalities, giving both heuristic and exact separation procedures. The computation results show that a three-index flow formulation is stronger than the two-index counterparts, however, this does not always lead to better algorithmic performance.

The first use of column generation as an approach for tackling the LRP is due to Berger et al. (2007). Here, the authors develop a Branch-and-Price algorithm to solve instances with uncapacitated facilities and routes limited by a maximum length. The authors report computational experiments on self-generated instances with 10 facilities and up to 100 customers. Some of these instances are solved to optimality within a running time of two hours. Akca et al. (2009) give a set-partitioning formulation for the standard capacitated version of LRP and solve it using a branch-and-price algorithm. The authors apply three distinct heuristics to price negative reduced cost columns, calling the exact labelling algorithm only when the heuristics fail to find such columns. They were able to solve instances with up to 5 facilities and 40 customers.

Based on the formulation of Akca et al. (2009), Baldacci et al. (2011b) propose an efficient lower-bounding procedure based on dynamic programming and dual ascent methods. Using these lower bounding procedures, the authors propose a decomposition of the LRP into a collection of CVRP-CMD instances. The solution strategy for the LRP consists of solving the CVRP-CMD for all the subsets of the facilities, and keeping the solution with the smallest cost. Baldacci et al. (2011b) show that a lower bounding procedure can significantly reduce the number of facility subsets, by identifying those subsets that cannot lead to optimal solutions. Their algorithm clearly outperforms the solution methods known at that time.

Inspired by Baldacci et al. (2011b), Contardo et al. (2014) develop a solution approach based on the enumeration of subsets of facilities. Starting with a given upper bound, they

use a Branch-and-Cut algorithm to solve the formulation proposed by Belenguer et al. (2011), strengthened by some valid inequalities introduced in Contardo et al. (2013a), while requiring integrality only on the facility opening variables. Then, for each subset of facilities that may lead to an optimal solution, the linear relaxation of the corresponding CVRP-CMD instance is solved by cut-and-column generation giving a valid lower bound on the LRP subject to opening these facilities. Finally, they enumerate all columns whose reduced costs are not greater that the gap between the upper and lower bounds for the current subset of facilities. Once all the columns are enumerated, they solve the standard set-partitioning formulation to determine the optimal integer solution, which is done using a standard MILP solver. When compared to the work of Baldacci et al. (2011b), this new approach is able to provide tighter lower bounds and it also solves to optimality two of the remaining open instances.

As observed by Schneider and Drexl (2017), both the algorithms proposed by Baldacci et al. (2011b) and Contardo et al. (2014) are very sophisticated and rely on a number of complex algorithmic and implementation refinements. Another point worth mentioning is that these methods exploit the fact that most of the instances then found in the literature have a rather small number of potential facilities. Hence the enumeration of all subsets is somewhat manageable by their approach. However, it is unlikely that these methods could be able to deal with larger instances, such as the new benchmarks introduced by Schneider and Löffler (2019), containing up to 600 customers and 30 depot locations.

The LRP is also a fruitful topic for the development of both heuristics and matheuristics. A majority of these methods works in a two-stage hierarchical fashion: in the first stage, the heuristics deal with the open-facility decisions. Once these decisions are made, the algorithms try to optimize the vehicle routing decisions. The detailed study of these approaches is beyond the scope of present work. Only to mention few, we highlight the works of Prins et al. (2006) where they introduce a GRASP with path relinking; Prins et al. (2007) which present a two-stage heuristic combining Lagrangian Relaxation and Granular Tabu Search (GTS); and Schneider and Löffler (2019) that propose a Tree Based Search Algorithm (TBSA) that explores the space of depot configurations in a tree-like fashion, and then, use a GTS to solve the multi-depot vehicle routing problem. We refer the reader to the survey of Schneider and Drexl (2017) for a complete overview of the techniques already employed in the context of LRP.

We now review the literature on some related problems with the nested knapsack structure. First of all, the CVRP-CDM subproblem of the LRP still has this structure. As mentioned above, this subproblem is considered by Baldacci et al. (2011b) and Contardo et al. (2014) when solving the LRP. To solve the CVRP-CDM, they adapt exact approaches

for standard multi-depot vehicle routing problem with uncapacitated depots by Baldacci and Mingozzi (2009) and Contardo and Martinelli (2014) In addition, Contardo et al. (2014) use valid inequalities proposed by Belenguer et al. (2011) and Contardo et al. (2013a). The CVRP-CDM is also encountered as a subproblem by Ben Mohamed et al. (2019) when solving the two-echelon stochastic multi-period capacitated location-routing problem using a logic-based Benders decomposition algorithm. To solve the CVRP-CMD, the authors use the package VRPSolver Pessoa et al. (2020), which extends the branch-cut-and-price algorithm by Sadykov et al. (2020) for a generic model that encompasses many VRP variants. Some pre-processing of the bounds on the number of routes that leave certain depots is performed in Ben Mohamed et al. (2019), but only valid inequalities known for the CVRP are used.

Dabia et al. (2019) introduced the capacitated vehicle routing problem with time windows and shifts (VRPTW-S). In this generalization of the standard vehicle routing problem with time windows, the time horizon is divided into non-overlapping shifts. Depending on when a route leaves from the depot, this route is assigned to one of the shifts. The total amount of freight delivered by routes belonging to a shift is limited by a loading capacity. Thus, the nested knapsack structure appears, as the demand of every customer contributes to the vehicle capacity and the shift loading capacity constraints. Dabia et al. (2019) propose a branch-cut-and-price algorithm for the VRPTW-S. Their main contribution is new cover inequalities for the problem. These inequalities are related to the RLKCs proposed in this chapter, as they are also derived from the higher-level knapsack inequalities (for the shift loading capacities), and they involve route variables. However, none of the families of cover inequalities proposed by Dabia et al. (2019) and RLKCs is the subset of the other.

Tilk et al. (2020) introduced the last-mile vehicle routing problem with delivery options (VRPDO), in which some requests can be shipped to alternative locations with possibly different time windows. Moreover, when delivery options share a common location, a locker for example, capacities must be respected when assigning shipments. Thus, we have here the double knapsack structure, as customer deliveries are subject to both vehicle and delivery location capacities. Knapsack constraints are however not nested: two customer deliveries by the same vehicle do not necessarily contribute to the same higher-level knapsack constraints corresponding to the delivery location capacities. Tilk et al. (2020) propose a branch-cut-and-price algorithm for the VRPDO which is similar to the one for the standard VRPTW except that they use a different graph to solve the pricing problem. No specific valid inequalities based on the knapsack structure of the problem are proposed.

Albareda-Sambola et al. (2009) introduced the capacity and distance constrained plant location problem. It is an extension of the discrete capacitated plant location problem, where the customers assigned to each plant have to be packed in groups that will be served by one vehicle each. The constraints include two types of capacity. On the one hand, plants are capacitated and the demands of the customers are indivisible. On the other hand, the total distance traveled by each vehicle to serve its assigned customers in round trips plant–customer–plant is also limited. This problem also has the nested knapsack structure. Here, however, different quantities contribute to the lower-level and higher-level knapsack constraints: plant–customer–plant distances to the former and customer demands to the latter. The authors propose integer programming formulations and a tabu search heuristic. Later, Fazel-Zarandi and Beck (2012) propose a logic-based Benders decomposition algorithm for this problem.

4.2 Problem definition and formulation

In this chapter and contrary to the two previous chapters, we denote as I the set of depots and as J the set of customers.

We now formally define the problem. The distribution network is represented by a weighted undirected graph $G = (I \cup J, E \cup F)$. Vertices in I represent depots, J denotes a set of customers. Edges $E = J \times J$ and $F = I \times J$ represent cheapest paths, with costs $c : E \cup F \to \mathbb{R}_+$, between pairs of vertices. Additionally, we associate capacities $W : I \to \mathbb{N}_+$ with depots, demands $d : J \to \mathbb{N}_+$ with customers, and an opening cost $d : I \to \mathbb{R}_+$ with depots. Depots have an unlimited fleet of homogeneous vehicle with capacity $Q \in \mathbb{Z}_+$.

In this context, a *route* is an elementary cycle in G containing exactly one depot in I and a subset of customers in J. A feasible solution is a set of routes such that: (i) each customer belongs to exactly one route; (ii) the sum of the demands of the customers in a route does not exceed Q; (iii) the sum of the demands of the customers in all routes associated to depot $i \in I$ does not exceed W_i. The goal is to find a feasible solution that minimizes the total route cost.

4.2.1 Formulation

Let Ω_i be the set of all routes leaving the depot $i \in I$ and that ship at most Q items. Given $I' \subseteq I$, we denote by $\Omega(I') = \bigcup_{i \in I'} \Omega_i$ the set of all routes incident to the depots in I'. The set of all possible routes is denoted by Ω.

Furthermore, given route $\omega \in \Omega$, let $\tilde{b}_e^\omega \in \{0, 1, 2\}$ be the number of times the edge $e \in E \cup F$ is traversed by the route ω and let $\tilde{a}_j^\omega = \frac{1}{2} \sum_{e \in \delta(j)} \tilde{b}_e^\omega$ be equal to 1 if customer $j \in J$ is served by route ω, 0 otherwise. Consequently, we define $\tilde{d}^\omega = \sum_{j \in J} \tilde{a}_j^\omega d_j$ as the load of the route and $\tilde{c}^\omega = \sum_{e \in E \cup F} \tilde{b}_e^\omega c_e$ as the cost of the route.

The *Capacitated Location-Routing Problem* (LRP) consists in finding a subset of depots $I' \subseteq I$ to open, and a collection of feasible routes $\Omega'(I') \subseteq \Omega(I')$, such that every customer of J is included in exactly one route $\omega \in \Omega'(I')$ and the sum of loads of all the routes leaving depot $i \in I'$ does not exceed its capacity W_i (lower-level knapsack constraints). The objective is to minimize the sum of opening costs and routing costs.

Let y_i be a binary variable equal to 1 if the depot $i \in I$ is opened, 0 otherwise. Let λ_ω be a binary variable equal to 1 if a vehicle uses route ω, 0 otherwise. We formulate the LRP as:

$$[\text{F}] \equiv \min \sum_{i \in I} f_i y_i + \sum_{\omega \in \Omega} \tilde{c}^\omega \lambda_\omega \tag{4.1}$$

$$\text{s.t.} \quad \sum_{\omega \in \Omega} \tilde{a}_j^\omega \lambda_\omega = 1 \qquad \forall\, j \in J \tag{4.2}$$

$$\sum_{\omega \in \Omega_i} \tilde{d}^\omega \lambda_\omega \leq W_i y_i \qquad \forall\, i \in I \tag{4.3}$$

$$\lambda_\omega \in \{0, 1\} \qquad \forall\, \omega \in \Omega, \tag{4.4}$$

$$y_i \in \{0, 1\} \qquad \forall\, i \in I \tag{4.5}$$

$$\tag{4.6}$$

Expression (4.1) is the objective. Constraints (4.2) ensure that each customer is visited only once. Constraints (4.3) are the higher-level knapsack constraints and make sure that the sum of the loads of the routes incident to a depot does not exceed the capacity of the depot. These constraints also ensure that if the route leaves a depot, then the depot is open. Finally, constraints (4.4) and (4.5) are domains of variables.

It is very straightforward to formulate the CVRP-CMD and the VRPTW with Shifts from the formulation (F) of the LRP. For the CVRP-CMD, since all depots are open, we just fix all variables y to 1. For the VRPTW with Shifts, we create a fictive depot for each shift. Since a vehicle can start from any shift at no cost, we fix all variables y to 1. Then, we introduce a time resource in the pricing problem like what we did in Chapter 3 for the 2E-VRPTW. The time window of a depot corresponds to the shift it is associated to. We also define time windows for customers.

4.2.2 Robust valid inequalities

Robust valid inequalities do not change the structure of the pricing problem as they can be expressed using only arc variables. These inequalities change only the reduced costs of the arcs. Some of these valid families of inequality have already been presented in previous chapters. These are the RCCs presented in chapter 1 and the VCIs presented in chapter 2. We also use the non-robust lm-R1Cs presented in 1.

To simplify the presentation, we use additional auxiliary variables. Let $x_e^i = \sum_{\omega \in \Omega_i} \tilde{b}_e^\omega \lambda_\omega$ be equal to 1 if the edge $e \in E \cup F$ is traversed by a route incident to depot $i \in I$. Let $z_j^i = \frac{1}{2} \sum_{e \in \delta(j)} x_e^i$ be equal to 1 if customer $j \in J$ is visited by a route starting at depot $i \in I$.

First family of valid inequalities is very straightforward. Since any depot chosen to be opened must be origin of at least one valid route, the following inequalities are valid for the LRP :

$$\sum_{e \in \delta(i)} x_e^i \geq 2y_i, \quad \forall\, i \in I. \tag{4.7}$$

Since this family contains a little number of inequalities, we do not separate them and add all of them to the formulation [F].

Let $o = \min \left\{ |I'| : I' \subseteq I, \sum_{i \in I'} W_i \geq \sum_{j \in J} d_j \right\}$ be a lower bound on the number of depots opened required to cover the demand of all customers. The following inequality is also valid for the LRP

$$\sum_{i \in I} y_i \leq o. \tag{4.8}$$

This inequality is added to the formulation [F].

Depot Capacity Cuts (DCC) This family of valid inequalities comes from Belenguer et al. (2011). If a subset if customers $J' \subset J$ cannot be served by a subset of depots $I' \subset I$, then at least one vehicle from a depot $i \in I \setminus I'$ should visit J'. The following valid inequalities are thus valid :

$$\sum_{i \in I \setminus I'} \sum_{e \in \delta(J')} x_e^i \geq 2 \qquad I' \subset I,\ J' \subseteq J \text{ such that } \sum_{j \in J'} d_j > \sum_{i \in I'} W_i. \tag{4.9}$$

Several successive lifting of the latter inequalities were performed by Belenguer et al. (2011) and Liguori (2019). Here, we use the following depot capacity cuts (DCC) as presented in Liguori (2019). Given $I' \subset I$ and $J' \subset J$, let $r(I', J') = \left\lceil \frac{\sum_{j \in J'} d_j - \sum_{i \in I'} W_i}{Q} \right\rceil$

be a lower bound on the number of vehicles starting from $I \setminus I'$ required to serve the demand of customers in J'. Consider a pair of depots $i, i' \in I'$, the following valid inequalities are valid :

$$\sum_{i \in I \setminus I'} \sum_{e \in \delta(J')} x_e^i \lambda_\omega \geq 2r(I', J')y_i + 2r(I' \setminus \{i\}, J')y_{i'} - 2r(I' \setminus \{i, i'\}, J')(1 - y_i - y_{i'}) \tag{4.10}$$

$$\sum_{i \in I \setminus I'} \sum_{e \in \delta(J')} x_e^i \geq 2r(I', J')y_{i'} + 2r(I' \setminus \{i'\}, J')y_i - 2r(I' \setminus \{i, i'\}, J')(1 - y_i - y_{i'}) \tag{4.11}$$

In table 4.1, we report the best lower bound depending on the value of variables y_i and $y_{i'}$. We note that both inequalities (4.10) and (4.11) have right-hand sides at most equal to the value of the lower bounds reported in 4.1. They are thus valid for the LRP.

y_i	$y_{i'}$	lb on nb. of vehicles
0	0	$r(I' \setminus \{i, i'\}, J')$
0	1	$r(I' \setminus \{i\}, J)$
1	0	$r(I' \setminus \{i'\}, J)$
1	1	$r(I', J')$

Table 4.1: Lower bound on the number of vehicles depending on the value of variables y_i and $y_{i'}$ for $i, i' \in I'$

These inequalities are separated using a heuristic that combines GRASP and local search described in Liguori (2019).

Cover Inequalities (COV) We use cover inequalities to strengthen knapsack inequalities (4.3). Using the auxiliary variables introduced above, we rewrite these constraints as :

$$\sum_{j \in J} d_c z_j^i \leq W_i \quad i \in I \tag{4.12}$$

Given a depot $i \in I$ and a subset $J' \subset J$ of customers such that $\sum_{j \in J'} d_j \geq W_i$, the following *cover inequality*

$$\sum_{j \in J'} z_j^i \leq |J'| - 1 \tag{4.13}$$

is valid for the LRP.

These inequalities are separated using the classic IP separation procedure as presented, for instance, in Wolsey (1998). Given a solution $(\bar{y}, \bar{z})$ to [F], the separation consists in

solving the following LP for each depot i such that $\bar{y}_i > 0.25$,

$$\min \left\{ \sum_{j \in J} (1 - \bar{z}_j^i)\beta_j \mid \sum_{j \in J} d_j \beta_j \geq W_i + 1,\ \beta \in \{0,1\}^{|J|} \right\} \tag{4.14}$$

where variable β_j equal 1 if customer j belongs to the cover J', 0 otherwise. If the value of the optimal solution to the IP is less than 1, then the cover inequality characterized by J' is violated.

Fenchel cuts (FCC) In a feasible solution to the LRP, the total capacity of the depots open must be larger than the total demand of the customers :

$$\sum_{i \in I} W_i y_i \geq \sum_{j \in J} d_j. \tag{4.15}$$

This covering constraint is strengthen using Fenchel cuts (FCC). These cuts are generated by a separation problem instead of trying to devise a family of valid inequalities for the problem. Separation of Fenchel cuts was studied by Boyd (1993), Boyd (1994), and used for instance by Boccia et al. (2008) to strengthen knapsack constraints. The goal of such a procedure is to separate valid inequalities that are valid for the convex hull of the solutions to the set-partitioning inequality considered.

Let $H = (h^q)_{q \in Q}$ be the set of the extreme points of the convex hull of

$$P_{\text{COV}} = \left\{ h \mid \sum_{i \in I} W_i h_i \geq \sum_{j \in J} d_j \ , h \in \{0,1\}^{|I|} \right\} \tag{4.16}$$

that contains all feasible solutions to the covering constraint. Let $\bar{y}$ be the solution to the linear relaxation of [F]. Let us now check if we can express $\bar{y}$ as a linear combination of solutions in H that satisfies the covering constraint *i.e.* find non-negative multipliers $(\chi_q)_{q \in Q}$ such that $\bar{y} = \sum_{q \in Q} h^q \chi_q$ and $\sum_{q \in Q} \chi_q \geq 1$. Therefore, we solve the following LP :

$$\max \left\{ \sum_{q \in Q} \chi_q \mid \sum_{q \in Q} h^q \chi_q \leq \bar{y}_i,\ i \in I \text{ and } \chi_q \geq 0,\ q \in Q \right\} \tag{4.17}$$

Let (χ^*, ψ^*) be the optimal primal and dual solutions to (4.17). Since strong duality applies, $\sum_{q \in Q} \chi_q^* = \sum_{i \in I} \bar{y}_i \psi_i^*$ holds. If $\sum_{q \in Q} \chi_q^* < 1$ and so $\sum_{i \in I} \bar{y}_i \psi_i^* < 1$, then we cannot express $\bar{y}$ as a linear combination of solutions to H that satisfies the covering constraint. Then, the inequality $\sum_{i \in I} \psi_i^* y_i \geq 1$ is valid of for P_{COV}. In our algorithm, we separate

the inequality by directly solving the dual problem of (4.17) :

$$\min \left\{ \bar{y}^{\intercal}\psi \mid \sum_{i \in I} h_i^q \psi_i \geq 1,\ q \in Q \text{ and } \psi_i \geq 0,\ i \in I \right\}. \tag{4.18}$$

Note that we can separate FCCs in such a way if we can enumerate extreme points of (4.16). Here, since $|I| \leq 20$, this is the case.

4.2.3 Route load knapsack cuts

Consider the so-called *master knapsack polytope*:

$$\mathcal{P}_{\text{MKP}}(W) = \text{conv} \left\{ t_q \in \mathbb{Z}_+^W : \sum_{q=1}^{W} q\, t_q = W \right\}.$$

Next theorem characterizes the knapsack (non-trivial) facets of this polytope.

Theorem 1 (Aráoz (1974))**.** *The coefficient vectors $\xi \in \mathbb{R}_+^W$ of the knapsack facets $\xi t \leq 1$ of $\mathcal{P}_{MKP}(W)$ with $\xi_1 = 0$ and $\xi_W = 1$ are the extreme points of the system of linear constraints*

$$\xi_1 = 0, \tag{4.19}$$
$$\xi_W = 1, \tag{4.20}$$
$$\xi_q + \xi_{W-q} = 1, \qquad \forall 1 \leq q \leq W/2, \tag{4.21}$$
$$\xi_q + \xi_{q'} \leq \xi_{q+q'}, \quad \forall q + q' < W. \tag{4.22}$$

The feasible solutions ξ to the system give valid inequalities $\xi t \leq 1$ for $\mathcal{P}_{MKP}(W)$.

Constraints (4.21) and (4.22) are *complementarity* and *superadditivity* constraints. Therefore, every coefficient vector ξ is a non-decreasing function because $\xi_q = 0 + \xi_q = \xi_1 + \xi_q \leq \xi_{q+1}$.

Let now θ_q^i be an integer variable indicating how many routes with a total load of *exactly* q units, where $1 \leq q \leq W_i$, leave depot $i \in I$. Variables θ can be expressed in terms of variables λ.

$$\theta_q^i = \sum_{\omega \in \Omega_i:\, \tilde{d}^\omega = q} \lambda_\omega, \quad \forall\, i \in I,\ 1 \leq q \leq W_i. \tag{4.23}$$

Then, the following inequalities are clearly valid for the formulation [F].

$$\sum_{q=1}^{W_i} q\,\theta_q^i \le W_i\,y_i, \quad i \in I, \tag{4.24}$$

$$\theta_q^i = 0, \qquad i \in I,\ Q < q \le W_i. \tag{4.25}$$

Inequalities (4.24) and (4.25) are redundant for the linear relaxation of the formulation [F]. However, they define an integer knapsack-like polyhedron for each depot i. Thus, they can be used as a source of non-redundant cuts, as shown by the next theorem.

Theorem 2. *Given a depot $i \in I$, let $\xi \in \mathbb{R}_+^{W_i}$ be a feasible solution to the system (4.19)–(4.22). Then the inequality*

$$\sum_{q=1}^{W_i} \xi_q\,\theta_q^i \le y_i \tag{4.26}$$

is valid for the formulation [F].

Proof. If $y_i = 0$, then we have $\theta_q^i = 0$ for all $1 \le q \le W_i$ due to (4.24). Thus, inequality (4.26) is satisfied.

Let now $y_i = 1$. Consider an arbitrary vector $\bar\theta^i \in \mathbb{Z}_+^{W_i}$ which satisfies (4.24)–(4.25). Let $q' = W_i - \sum_{q=1}^{W_i} q\,\bar\theta_q^i$. Consider vector $\widetilde\theta^i \in \mathbb{Z}_+^{W_i}$ that is defined by $\widetilde\theta_q^i = \bar\theta_q^i$ for all $1 \le q \le W_i$ such that $q \ne q'$, and $\widetilde\theta_q^i = \bar\theta_q^i + 1$ if $q = q'$. We have $\bar\theta^i \le \widetilde\theta^i$ and $\widetilde\theta^i \in \mathcal{P}_{\mathrm{MKP}}(W_i)$. By Theorem 1, $\bar\theta^i$ satisfies (4.26), and we have

$$\sum_{q=1}^{W_i} \xi_q\bar\theta_q^i \le \sum_{q=1}^{W_i} \xi_q\widetilde\theta_q^i \le y_i.$$

Thus, $\bar\theta^i$ also satisfies (4.26). □

Valid inequalities can also be obtained by Chvàtal-Gomory rounding of constraints (4.24). They can be rewritten in the following form.

$$\sum_{\omega\in\Omega_i} \tilde d^\omega \lambda_\omega \le W_i y_i, \quad i \in I. \tag{4.27}$$

By applying Chvàtal-Gomory rounding of constraints (4.27) with a multiplier $\beta \in \mathbb{R}$, where $\beta \ge 1/W_i$, we obtain inequalities

$$\sum_{\omega\in\Omega_i)} \frac{\lfloor \beta\,\tilde d^\omega \rfloor}{\lfloor \beta\,W_i \rfloor}\lambda_\omega \le y_i, \quad i \in I, \tag{4.28}$$

which are valid for the formulation [F].

We now define a family of *Route Load Knapsack Cuts* (RLKC), which forms the union of inequalities (4.26) and (4.28). Each RLKC is characterized by a depot $i \in I$ and a non-decreasing and superadditive function $g(q)$ defined for all values $0 < q \leq Q$:

$$\sum_{\omega \in \Omega_i} g(\tilde{d}^\omega)\lambda_\omega \leq y_i. \tag{4.29}$$

For constraints (4.26), function $g(q) = \xi_{\lfloor q \rfloor}$ for $1 \leq q \leq Q$, and $g(q) = 0$ if $0 < q < 1$. This function is non-decreasing and superadditive by definition of vector ξ. For constraints (4.28), function $g(q)$ is defined as $\lfloor \beta\, q \rfloor / \lfloor \beta\, W_i \rfloor$. This function is obviously non-decreasing. Superadditivity comes from the fact that $\lfloor q \rfloor + \lfloor q' \rfloor \leq \lfloor q + q' \rfloor$ for all $q, q' \in \mathbb{R}_+$. We exploit monotonicity and superadditivity of function $g(q)$ in the pricing algorithm below. Note, however, that inequalities (4.29) are not necessarily valid for the formulation [F] for every non-decreasing and superadditive function $g(q)$.

4.2.4 Separation of Route load knapsack cuts

Separation of RLKCs is done separately for each depot $i \in I$. For clarity, we omit index i for the remaining of this section: we have $\Omega = \Omega_i$, $\theta = \theta^i$, $W = W_i$. Let $\bar{\lambda}$ be a solution to the linear relaxation of the formulation [F]. Let $\bar{\Omega}$ be the set of routes participating in solution $\bar{\lambda}$: $\bar{\Omega} = \{\omega \in \Omega : \bar{\lambda}_\omega > 0\}$. Let also $\bar{\theta}_q = \sum_{\omega \in \bar{\Omega}:\, \tilde{d}^\omega = q} \bar{\lambda}_\omega$.

First, we separate Route Load Knapsack Cuts by Chvàtal-Gomory rounding, i.e. constraints (4.28). We enumerate all distinct multipliers β such that there exists $\omega \in \bar{\Omega}$ for which $\beta\, \tilde{d}^\omega$ is integer. For each such multiplier β, we check whether constraint (4.28) is violated by $\bar{\lambda}$.

Secondly, we separate Route Load Knapsack Cuts in the form (4.26). It would be possible to perform an exact separation, by solving the following LP:

$$z = \max \left\{ \sum_{i=1}^{W} \bar{\theta}_q \xi_q \text{ subject to (4.19)–(4.22)} \right\}.$$

If $z > y_i$, then inequality (4.26) would be violated. However, as there are W variables and $O(W^2)$ constraints in that LP, that would be too time-consuming for large values of W. Also, generation of only one violated cut would be undesirable due to possible convergence issues.

Therefore, we separate only the cuts that correspond to so-called $1/k$-facets of the master knapsack polytope. A knapsack (non-trivial) facet $\xi t \leq 1$ is called an $1/k$-*facet* if

k is the smallest possible integer such that

$$\xi_q \in \{0/k,\ 1/k,\ 2/k, \ldots, k/k\} \cup 1/2, \quad 1 \leq q \leq W. \tag{4.30}$$

Using both theoretical arguments and computational experiments, it was showed in Chopra et al. (2015) that the $1/k$-facets for small values of k are the most important facets for obtaining a good approximation to $\mathcal{P}_{\text{MKP}}(W)$.

A sequence $\xi = (\xi_q)_{q=1}^{W}$ is called *symmetric* if the complementarities (4.21) hold. We call $\xi t \leq 1$ an $1/k$-*inequality* if ξ is a non-decreasing symmetric sequence that satisfies (4.19)–(4.20) and (4.30). In general, a $1/k$-inequality need not be a valid inequality for $\mathcal{P}_{\text{MKP}}(W)$. A $1/k$-inequality is uniquely determined by a non-decreasing sequence (a_m) where a_m represents the first index q with $\xi_q \geq m/k$ for $m \in \{1, \ldots, k\} \cup \{k/2\}$ (a_0 is not part of the sequence because it would always have value 1). Such a sequence ξ will be denoted by $\xi^{k-(a_m)}$. The next theorem describes the necessary relationships between the elements of the sequence (a_m) for defining the coefficients of a $1/k$-inequality $\xi^{k-(a_m)}$ that is valid for $\mathcal{P}_{\text{MKP}}(W)$.

Theorem 3 (Chopra et al. (2015))**.** *An $1/k$-inequality $\xi^{k-(a_m)}$ satisfies (4.19)–(4.22) if and only if*

$$2 \leq a_m \leq a_{m'} \leq (W+1)/2, \quad \forall m < m' \leq k/2, \tag{4.31}$$

$$a_m + a_{k+1-\lceil m \rceil} = W + 1, \quad \forall m \leq k/2, \tag{4.32}$$

$$a_m + a_{m'} \geq a_{\lceil m+m' \rceil} \quad \forall m \leq m' \text{ with } \lceil m + m' \rceil \leq k. \tag{4.33}$$

In our separation algorithm, we enumerate non-decreasing sequences (a_m) satisfying constraints (4.31)–(4.33) that correspond to 1/6-, 1/8-, or 1/10-inequalities. Let Φ be the set of all possible loads of routes participating in the fractional solution: $\Phi = \{q : 1 \leq q \leq W,\ \bar{\theta}_q > 0\}$.

To separate 1/6-inequalities $\xi^{6-(a_m)}$, we enumerate triples (a_1, a_2, a_3) such that $a_m \in \Phi$ or $\{W - a_m\} \in \Phi$ for $m = 1, \ldots 3$, and $2 \leq a_1 \leq a_2 \leq a_3 \leq (W+1)/2$. Values a_4, a_5, a_6 are then obtained from equalities (4.32). According to constraints (4.33), for each triple (a_1, a_2, a_3), we verify $2a_1 \geq a_2$, $a_1 + a_2 \geq a_3$, $a_1 + 2a_3 \geq W + 1$ (obtained from $a_1 + a_3 \geq a_4$ or from $a_3 + a_3 \geq a_6$), and $2a_2 + a_3 \geq W + 1$ (obtained from $a_2 + a_3 \geq a_5$ or from $a_2 + a_2 \geq a_4$).

To separate 1/8-inequalities $\xi^{8-(a_m)}$, we enumerate tuples (a_1, a_2, a_3, a_4) such that $a_m \in \Phi$ or $\{W - a_m\} \in \Phi$ for $m = 1, \ldots 4$, and $2 \leq a_1 \leq a_2 \leq a_3 \leq a_4 \leq (W+1)/2$. Values a_5, a_6, a_7, a_8 are then obtained from equalities (4.32). According to constraints (4.33),

for each tuple (a_1, a_2, a_3, a_4) we verify $2a_1 \geq a_2$, $a_1 + a_2 \geq a_3$, $a_1 + a_3 \geq a_4$, $2a_2 \geq a_4$, $a_1 + 2a_4 \geq W + 1$, $a_2 + a_3 + a_4 \geq W + 1$, and $3a_3 \geq W + 1$.

To separate 1/10-inequalities $\xi^{10-(a_m)}$, we enumerate tuples $(a_1, a_2, a_3, a_4, a_5)$ such that $a_m \in \Phi$ or $\{W - a_m\} \in \Phi$ for $m = 1, \ldots 5$, and $2 \leq a_1 \leq a_2 \leq a_3 \leq a_4 \leq a_5 \leq (W + 1)/2$. Values a_6, a_7, a_8, a_9, a_{10} are then obtained from equalities (4.32). According to constraints (4.33), for each tuple $(a_1, a_2, a_3, a_4, a_5)$ we verify $2a_1 \geq a_2$, $a_1 + a_2 \geq a_3$, $a_1 + a_3 \geq a_4$, $a_1 + a_4 \geq a_5$, $2a_2 \geq a_4$, $a_2 + a_3 \geq a_5$, $a_1 + 2a_5 \geq W + 1$, $a_2 + a_4 + a_5 \geq W + 1$, $a_3 + 2a_4 \geq W + 1$, and $2a_3 + a_5 \geq W + 1$.

For each sequence (a_m) which verifies (4.31)–(4.33), we generate sequence $\xi^{k-(a_m)}$, and check if the corresponding inequality (4.26) is violated by $\bar{\theta}$. If a positive violation is found, the inequality is added to the restricted master problem. Some additional remarks on this separation procedure:

- A separated valid $1/k$- inequality, defined by $\xi^{k-(a_m)}$, is not necessarily a facet (it is if and only if that vector is an extreme point of (4.19)–(4.22)). Yet, repeated separation rounds until no violation is found leads to exactly the same bounds that would be obtained by only separating $1/k$-facets.
- The enumerative approach used in the procedure is likely to be effective even for large values of W because of the typical sparsity of the fractional solution vector (if W is large, usually $|\Phi| << W$).
- An $1/k'$-inequality is also an $1/k$-inequality if k' is a divisor of k. So, 1/2-, 1/3-, 1/4-, and 1/5-inequalities are also being separated by the procedure.

4.3 Pricing routes

Given a dual solution to the linear relaxation of the restricted formulation [F], the pricing problem searches for variables λ_ω with a negative reduced cost. The pricing problem can be decomposed into $|I|$ similar subproblems, one for each depot $i \in I$. In this section, we consider a pricing subproblem for a fixed depot i.

When there is neither active rank-1 cuts (1.26) nor road load knapsack cuts (4.29), the reduced cost of variable λ_ω corresponding to route $\omega \in \Omega_i$ can be expressed as the sum of reduced costs for every edge in the graph. The reduced cost of an edge $e \in E \cup F$ is equal to the difference between the edge cost c_e and the current dual of the auxiliary variables x_e^i.

The pricing problem then can be formulated as a resource-constrained elementary shortest path problem (RCSPP) in the complete directed graph $D^i = (\{i\} \cup J, A)$. The

reduced cost $\bar{c}_a$ of every arc $a \in A$ in this graph D^i is equal to the reduced cost $\bar{c}_e$ of the corresponding edge e in graph G. We have a single capacity resource. The resource consumption of arc $(j, j') \in A$ is equal to $\frac{1}{2}d_j + \frac{1}{2}d_{j'}$ considering that $d_i = 0$. Bounds on the accumulated resource consumption are given on vertices. These bounds are $[0, Q]$ for every node $j \in \{i\} \cup J$. The RCSPP seeks to find an elementary cycle of minimum reduced cost starting and finishing in node i.

The main goal of this section is to describe the modifications of the labelling algorithm for the case with active road load knapsack cuts (4.29). The coefficient of variable λ_ω in an active RLKC $\gamma \in \Gamma$ is equal to the value of a non-decreasing and superadditive function $g_\gamma(\tilde{d}^\omega)$ of the route load $\tilde{d}^\omega$. Thus, the total contribution of active RLKC to the reduced cost of route $\omega \in \Omega_i$ (taking into account that dual values for these cuts are non-positive) is a linear combination of functions $g_\gamma(\tilde{d}^\omega)$ for all $\gamma \in \Gamma$. Such a linear combination $g(\tilde{d}^\omega)$ is also a non-decreasing and superadditive function.

In the following, we describe the modifications to the label extension procedure, dominance relations, and the definition of completion bounds.

In the labelling algorithm, each label L represents a partial path $\omega(L)$ from the node i. Let $J(L)$ be the set of customers visited by the partial path, $j(L)$ be the final node of the partial path, $\bar{c}(L)$ be the sum of reduced costs of arcs in the partial path, and $q(L)$ be the total capacity resource consumption of the partial path. The algorithm consists in an enumeration of all feasible partial paths. For that, every label L is extended by taking each arc $a' = (j(L), j')$ outgoing from node $j(L)$. After extension, a new label L' is created, for which, $j(L') = j'$, $q(L') = q(L) + \frac{1}{2}d_{j(L)} + \frac{1}{2}d_{j(L')}$, and $\bar{c}(L') = \bar{c}(L) + \bar{c}_{a'}$.

To avoid complete enumeration, a dominance rule is used to remove labels which cannot lead to the minimum reduced cost path when extended. A label L dominates label L' if for any partial path ω such that union of paths $\omega(L')$ and ω is feasible, i.e. $(\omega(L'), \omega) \in \Omega$, we have : (i) union of paths $\omega(L)$ and ω is also feasible and (ii) reduced cost of path $(\omega(L'), \omega)$ is not smaller than the reduced cost of path $(\omega(L), \omega)$. The next theorem gives a valid dominance rule.

Theorem 4. *Given a non-decreasing function $g(\tilde{d}^\omega)$ representing the contribution of the route load knapsack cuts to the reduced cost of a path ω_Ω, label L dominates label L' if $j(L) = j(L')$, $J(L) \subseteq J(L')$, $q(L) \leq q(L')$, and $\bar{c}(L) \leq \bar{c}(L')$.*

Proof. Consider an arbitrary partial path ω starting at node $j(L) = j(L')$ and finishing at node i. If path $(\omega(L'), \omega)$ is feasible then path $(\omega(L), \omega)$ is also feasible due to conditions $j(L) = j(L')$, $J(L) \subseteq J(L')$, and $q(L) \leq q(L')$. The reduced cost of path $(\omega(L'), \omega)$ is not smaller than one of $(\omega(L), \omega)$, as the total reduced costs of arcs of the former is

not smaller due to $\bar{c}(L) \leq \bar{c}(L')$, and the contribution of RLKCs to the reduced cost of $(\omega(L'), \omega)$ is not smaller than the one of $(\omega(L), \omega)$ due to $q(L) \leq q(L')$. Thus, L dominates L'. □

The forward-backward route symmetry is exploited in the bi-directional labelling algorithm in the following way. Every label L is extended only if $q(L) \leq Q/2$. After the label extension phase, the concatenation phase is performed, in which partial paths corresponding to two generated labels are concatenated to form a complete path. To speed up this concatenation phase, *completion bounds* are used. Given a node $j \in J$ and a set of labels $\mathcal{L}$ such that $j(L) = j$ for all $L \in \mathcal{L}$, completion bound $B_1(j, \mathcal{L})$ gives the minimum reduced cost of labels in $\mathcal{L}$: $B_1(j, \mathcal{L}) = \min_{L \in \mathcal{L}}\{\bar{c}(L)\}$. Due to the fact that function $g(q)$ is non-decreasing, value

$$\bar{c}(L') + g(q(L')) + B_1(j, \mathcal{L}) \tag{4.34}$$

gives a valid lower bound for the reduced cost of any complete path obtained by concatenation of paths $\omega(L')$ and $\omega(L)$ with $j(L') = j$ and $L \in \mathcal{L}$. If value (4.34) is not smaller than the minimum reduced cost of a complete feasible path found so far, then concatenation of label L' with all labels in $\mathcal{L}$ may be skipped.

However, completion bounds $B(j, \mathcal{L})$ may not be tight as the reduced cost of labels does not include the contribution of RLKCs. We can reinforce completion bounds B_1 by defining $B_2(j, \mathcal{L}) = \min_{L \in \mathcal{L}}\{\bar{c}(L) + g(q(L))\}$. The total load of any concatenated path $(w(L'), w(L))$, where $j(L') = j(L)$, is equal to $q(L) + q(L')$. Thus $g(q(L) + q(L'))$, i.e. the contribution of RLKCs to the reduced cost of this path, is not smaller than $g(q(L)) + g(q(L')$ due to superadditivity of function $g(q)$. Therefore, value (4.34) in which B_1 is replaced by B_2 is still a valid lower bound for the reduced cost of any complete path obtained by concatenation of paths $\omega(L')$ and $\omega(L)$ with $j(L') = j$ and $L \in \mathcal{L}$.

In the BCP approach, completion bounds are used not only in the concatenation phase of the labelling algorithm, but also during the bucket arc elimination procedure, as well as in the elementary route enumeration procedure. After completing the label extension phase, completion bounds are computed for all nodes $j \in J$ and certain values q'. Given a pair (j, q'), set $\mathcal{L}$ includes all labels L such that $j(L) = j$ and $q(L) \leq q'$.

4.4 Computational experiments

The implementation of the proposed algorithm is done in Julia 0.6, using JuMP package Dunning et al. (2017). The separation algorithm for the RLKCs and the modification

of the labelling algorithm for the case with active RLKCs are implemented in C++ language. We use the following third-party libraries and codes :

- BaPCod C++ library (Vanderbeck et al., 2019) which implements the BCP framework
- C++ code, developed by Sadykov et al. (2020) which implements the bucket graph based labelling algorithm, bucket arc elimination procedure, elementary route enumeration, and the separation algorithm for R1Cs
- CVRPSEP C++ library (Lysgaard, 2018) which implements heuristic separation of RCCs
- IBM CPLEX Optimizer version 12.10 as the LP solver in column generation

Experiments are run on a 2 Dodeca-core Haswell Intex Xeon E5-2680 v3 servers at 2.5 GHz. Every server has 128 Go of RAM. Each instance is solved using a single thread.

4.4.1 Location-routing instances

In this section we test our algorithm on literature instances of the LRP. The first set of instances, which we denote as `PPC06`, is introduced by Prins et al. (2006). It contains 30 instances with 20, 50, 100, or 200 customers, and with 5 or 10 possible depot locations. We do not consider easy instances with 20 customers. Therefore, it remains 26 instances. The second set of instances, which we denote as `TB99`, is introduced by Tuzun and Burke (1999). In this instances, depots are uncapacitated. We consider only nine instances with 100 or 150 customers and 10 possible depot locations: P111112, P111212, P112112, P112212, P113112, P113212, P131212, P131112, P131212, and P132112. We chose theses instances because they are also used in Contardo et al. (2014). The third set of instances which we denote as `SL19` is introduced by Schneider and Löffler (2019). We consider instances with 100, 200, or 300 customers; and with 5, 10, 15, or 20 possible depot locations.

In the first experiment, we computationally evaluate the impact of different families of cuts on the efficiency of our algorithm. The evaluation is performed on classic LRP instances `PPW06`. We test the following variants of the BCP algorithm :

- BCP_0 — the base variant which can be considered as a straightforward application of the algorithm from Sadykov et al. (2020) for the LRP. Only two families of generic cuts are separated: rounded capacity cuts (RCC) and limited-memory rank-1 cuts (lm-R1C).

- BCP_{best} — the best variant in which all families of cuts are used. In addition to RCCs and lm-R1Cs, we separate problem-specific RLKCs, DCCs, FCCs, COVs, and VCIs.

- $\text{BCP}_{\text{best - CUT}}$ — the best variant, but without using one family of cuts CUT, CUT where CUT can be either RLKC, DCC, FCC, COV, or VCI.

To exclude as much as possible the randomness related to improving primal solutions, all algorithm variants are given the initial primal bound equal to the value of the best-known solution (among solutions found by us and solutions known in the literature). The time limit is set to 12 hours.

Table 4.2 gives an overview of the performance of seven variants of our algorithm. The columns give the variant, average primal-dual relative gap after solving the root node, the geometric mean of the time needed for solving the root node (in seconds), the average number of branch-and-bound nodes, the geometric mean of total solution time in seconds, and the number of instances solved within the time limit. The total solution time is equal to the time limit for the unsolved instances.

In Figure 4.1, we also give the performance profile for all the tested variants of the BCP algorithm. The horizontal axis is logarithmic. Given a line corresponding to a variant, each point (X, Y) of the line says that for Y instances the solution time for this variant is not more than X times larger than the minimum solution time for all variants. So, the higher is the line corresponding to a variant, the more efficient is this variant.

Table 4.2: Comparison of variants of the BCP algorithm on LRP instances `PPW06`

	Root				
Variant	Gap	Time (s)	Nodes	Time (s)	Solved
BCP_{0}	4.77%	96.9	18.6	1021.1	22/26
$\text{BCP}_{\text{best - RLKC}}$	0.53%	310.5	4.0	596.5	24/26
$\text{BCP}_{\text{best - DCC}}$	0.90%	181.1	7.9	735.9	22/26
$\text{BCP}_{\text{best - FCC}}$	0.71%	294.7	4.5	660.5	25/26
$\text{BCP}_{\text{best - COV}}$	0.52%	313.0	4.1	596.8	25/26
$\text{BCP}_{\text{best - VCI}}$	3.79%	227.3	8.3	888.7	25/26
BCP_{best}	0.52%	295.0	3.9	556.6	25/26

We see from the results that adding all problem-specific cuts makes the BCP algorithm significantly more efficient than the base variant. Among individual families of cuts, DCCs have the most impact on the number of instances solved to optimality. VCIs have the most impact on the average root gap and the mean total solution time. The impact of the family FCC is smaller. The families COV and RLKC have a very small impact

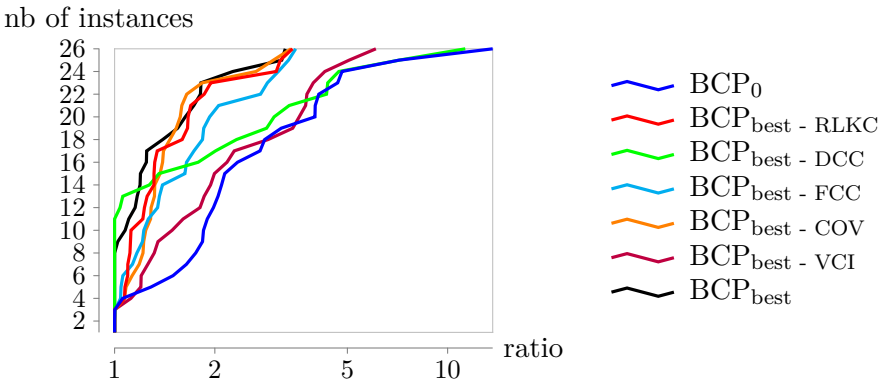


Figure 4.1: Performance profiles of BCP variants tested against Prodhon instances

both on the root gap and on the total solution time. However, the family of RLKCs allows us to solve one more instance to optimality.

In the next experiment, we compare our algorithm BCP_{best} with the algorithm by Contardo et al. (2014) on instances `TB99` and `PPW06`. For a fair comparison, we set the initial primal bounds to the same values as employed in Contardo et al. (2014). We set the time limit to 30 hours. No overall time limit is set for the approach of Contardo et al. (2014). The solution time for unsolved instances varies from around 6 hours to around 65 hours. The comparison is presented in Table 4.3. For each algorithm, we give the geometric mean time in seconds and the number of instances solved to optimality. The solutions times of Contardo et al. (2014) are normalized according to the difference of the processors used by them and by us.

Table 4.3: Comparison of BCP_{best} with Contardo et al. (2014) on instances in the sets `TB99` and `PPW06`

	BCP_{best}		Contardo et al. (2014)	
Instance set	Solved	Time (s)	Solved	Time (s)
`PPW06`	23/26	1103	16/26	1218
`TB99`	8/9	1219	6/9	7364

We see in Table 4.3 that our algorithm is significantly more efficient. It can solve to optimality 9 instances more than the approach by Contardo et al. (2014). One more instance `ppw200-10-1b` can be solved by our algorithm in less than 10 hours using the best-known solution in the literature as the initial upper bounds. Another instance `ppw200-10-2` is solved to optimality in less than 12 hours using the optimal solution value as the initial upper bound, as it can be seen from the results, presented in Table 4.2. The only remaining instance in the set `PPW06` is `ppw200-10-3b`. For two instances in the set `PPW06` we have improved the best-known solutions.

Finally, we experimentally evaluate the performance of our algorithm on recently introduced instances `SL19`. This is the first exact algorithm applied to these instances. We use here as initial primal bounds the best-known solutions found by Schneider and Löffler (2019). The time limit is set to 30 hours. The instances are divided into groups depending on their size, i.e. number of potential depot locations $|I|$ and the number of customers $|J|$. The results for each group of instances are shown in Table 4.4. The columns give the size of instances, the number of instances solved to optimality, the number of instances for which we find better feasible solutions than the best-known, and the average improvement for these instances.

Table 4.4: Performance of BCP_{best} on instances in the set `SL19`

Instances				
$\|I\|$	$\|J\|$	Solved	Improved BKS	Improvement
5	100	14/14	7/14	0.05%
10	100	13/14	5/14	0.11%
10	200	7/14	8/14	0.09%
15	200	10/20	13/20	0.12%
15	300	2/20	3/20	0.52%
20	300	3/20	3/20	0.48%

We see from Table 4.4 that our algorithm could solve to optimality all but one instance with 100 customers, about half of instances with 200 customers, and 15% of instances with 300 customers. Thus, we can say that instances in set `SL19` are more difficult than other literature instances. We could improve the best known solutions for 39 instances. The average improvement is very small for instances with up to 200 customers. The improvement becomes significant for instances with 300 customers. The results show that the heuristic of Schneider and Löffler (2019) has excellent quality for instances with up to 200 customers. This quality decreases for instances with 300 customers. The root gap (from the best-known solution) of our BCP algorithm for the largest instances sometimes reaches 6–8%. Whereas for instances solved to optimality a typical root gap is below 0.5% and never exceeds 2%. This shows that some best-known solutions for instances with 300 customers may be far away from optimum ones.

4.4.2 CVRP-CMD instances

In this section, we test our algorithm against CVRP-CMD instances which arise when solving the cut generation subproblem of the two-echelon stochastic multi-period capacitated location-routing problem by a logic-based Benders decomposition approach

Ben Mohamed et al. (2019). We have selected 199 instances of different difficulty, which have 50 customers and from three to five depots.

In this experiment, we computationally estimate the impact of valid inequalities on the efficiency of the BCP algorithm. In the CVRP-CMD, all depots are considered open. Therefore, variables y are fixed to one and valid inequalities FCC, COV, and VCI are not useful. Thus, we test the following BCP variants.

- BCP_0 — the base variant which can be considered as a straightforward application of the algorithm from Sadykov et al. (2020) for the CVRP-CMD. Again, only RCCs and lm-R1Cs are separated.
- $\text{BCP}_{0+\text{RLCK}}$ — the base variant with additional separation of RLKCs.
- $\text{BCP}_{0+\text{DCC}}$ — the base variant with additional separation of DCCs.
- BCP_{best} — the variant with separation of all valid inequalities (RCC, lm-R1C, DCC, and RLKC).

In this experiment, to exclude the randomness related to updating upper bounds, we use only instances that we were able to solve to optimality during preliminary tests. Thus, only 184 instances from 199 available are used. The initial upper bound is set to the optimum solution value augmented by a small epsilon. The time limit is set to 3 hours. Table 4.5 presents the results for each of the four BCP variants tested. The columns give the average relative root gap from the optimum solution value, the geometric mean value for the root solution time, the average number of branch-and-bound nodes, the geometric mean value for the total solution time, and the number of instances solved.

Table 4.5: Comparison of variants of the BCP algorithm on CVRP-CMD instances

	Root				
Variant	gap	time	Nodes	Time (s)	Solved
BCP_0	6.96%	18.4	122.6	1262	150/184
$\text{BCP}_{0+\text{RLCK}}$	6.70%	25.7	52.1	1022	175/184
$\text{BCP}_{0+\text{DCC}}$	4.89%	50.5	45.4	877	171/184
BCP_{best}	4.51%	71.4	24.6	754	180/184

In addition to Table 4.5, we also give the performance profiles for the four tested BCP variants in Figure 4.2. Each line corresponds to one variant of the algorithm and depicts the number of instances solved within a given time expressed in minutes.

Table 4.5 shows that both families DCC and RLKC have a positive and significant impact on the efficiency of the BCP algorithm. DCCs decrease the most the root gap,

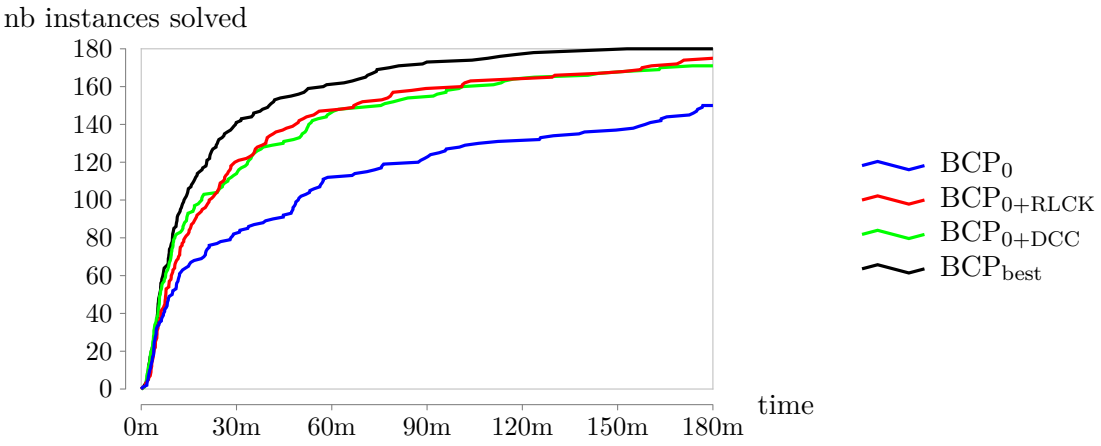


Figure 4.2: Performance profile for BCP variants tested against the CVRP-CMD instances

the number of nodes, and the average solution time. However, RLKCs have a larger impact on the number of solved instances. Clearly, the best variant of the BCP algorithm is the one which uses both families of cuts.

Among the 15 instances optimized using BCP_{best} with the best-known solution values as initial upper bounds, two of them could be solved to optimality, and 13 instances remain open. Although the CVRP-CMD is very similar to the standard multi-depot vehicle routing problem (MDVRP), the introduction of capacities to the depot makes the problem much more difficult. Literature MDVRP instances with 80 customers and less are consistently solved to optimality in seconds in the root node by Sadykov et al. (2020). Whereas CVRP-CMD instances we consider here are usually solved in 10–15 minutes and require many branch-and-bounds nodes to be explored. Moreover, 13 instances cannot be solved to optimality in three hours, even using the best-known solution values as the initial upper bounds. Root gaps are very large and may reach 15% of the optimum value.

4.4.3 VRPTW with Shifts instances

In this section, we test our BCP algorithm on the literature instances of the VRPTW with Shifts. These instances were introduced by Dabia et al. (2019) who built them on top of well-known Solomon instances for the VRPTW. There are instances with three different sizes: 25, 50, and 100 customers. All instances have three shifts. For each original Solomon instance, three instances were generated with three different shift capacities. For each size and each shift capacity there are 56 instances, divided into classes `c1`, `c2`, `r1`, `r2`, `rc1`, `rc2`. Thus, in total there are 504 instances.

Shifts in this problem are modeled as capacitated depots. Again, as in the case of the CVRP-CMD, there is no fixed cost to "open" a shift. Therefore, variables y are fixed

to one and valid inequalities FCC, COV, and VCI inequalities are not useful. Thus, as in the previous section, we test four BCP variants: BCP_0, BCP_{0+RLCK}, BCP_{0+DCC}, and BCP_{best}. We run these variants of our algorithm with the best-known solution values in the literature as initial upper bounds. The time limit is set to three hours.

Table 4.6 presents the results. The columns give the average relative root gap, geometric mean root solution time in seconds, the average number of branch-and-bound nodes, the geometric mean total solution time in seconds, and the number of instances solved. We also give the performance profiles for the four tested BCP variants in Figure 4.3. Each line corresponds to one variant of the algorithm and depicts the number of instances solved within a given time expressed in minutes.

Table 4.6: Comparison of variants of the BCP algorithm on VRPTW-S instances

	Root				
Variant	Gap	Time (s)	Nodes	Time (s)	Solved
BCP_0	4.14%	26.4	4.9	79.1	412
BCP_{0+RLCK}	1.62%	32.8	2.1	57.7	424
BCP_{0+DCC}	2.79%	38.7	3.8	90.1	417
BCP_{best}	1.31%	42.6	2.2	70.4	431

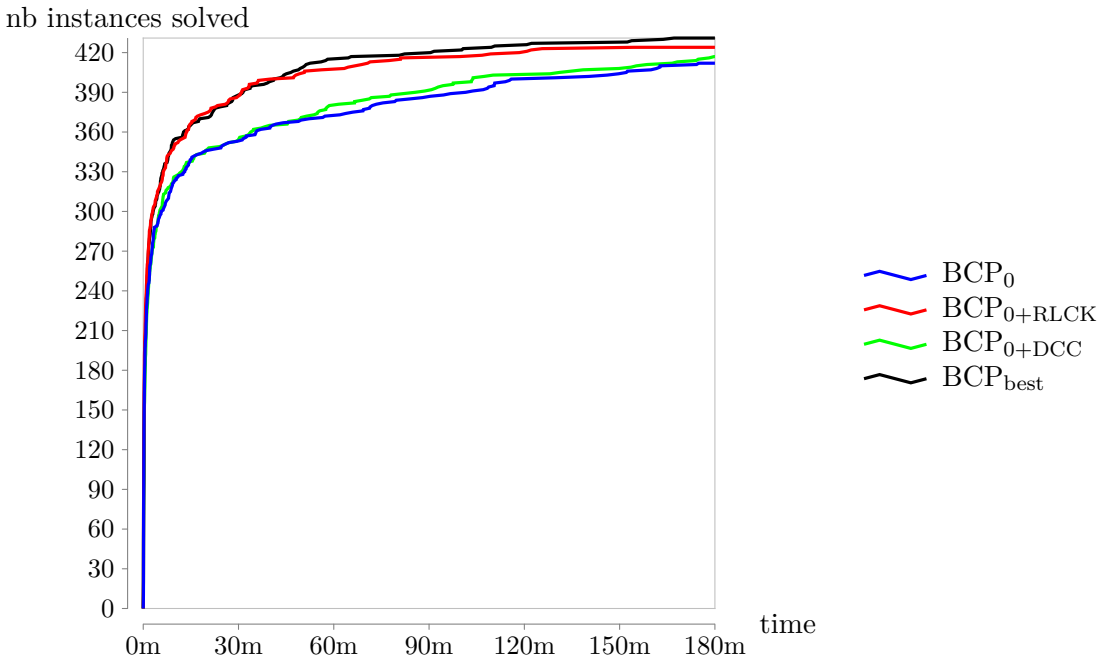


Figure 4.3: Performance profile for BCP variants tested against the VRPTW-S instances

The results show that, although the separation of DCCs reduces the root gap, their impact on the overall efficiency of the BCP algorithm is quite limited. On the other hand,

the impact of RLKCs is much more significant, as their separation improves the solution time, the number of branch-and-bound nodes, and the number of solved instances. On the performance profile, we also see that the impact of RLKCs is much larger than the one of DCCs. Both families of cuts have some cumulative effect, as the variant BCP_{best} solves the largest number of instances. This number (431 instances) is much larger than the number of instances solved to optimality by Dabia et al. (2019). We cannot however directly compare the two approaches as we used different time limits and different initial upper bounds. The best-known solutions are improved for 189 out of 504 instances.

Conclusion and Perspectives

In this thesis, we proposed and evaluated the performance of exact methods of mathematical optimization on several variants of vehicle-routing problems with two levels of decision. These problems arise from city logistics. They are hot topics in the context of increasing modes of freight transportation, growing flows of goods to and within cities, and restricting access to trucks of the city centers of major cities.

In Chapter 2, we proposed an improved branch-cut-and-price for the two-echelon capacitated vehicle routing problem (2E-CVRP). Our BCP algorithm includes techniques proposed quite recently for classic vehicle routing problems such as rounded capacity cuts, limited-memory rank-one cuts, ng-path relaxation, enumeration of elementary routes, and bucket-graph based labelling algorithms. Our algorithm also includes new components specific to the problem such as a new route based formulation without freight flow variables at satellites, a family of satellites supply inequality which can be interpreted as an adaptation of rounded capacity cuts and depot capacity cuts, and a new branching strategy on the number of urban trucks supplying each satellite that significantly reduces the size of the branch-and-bound tree.

We showed that our algorithm is highly efficient. Indeed, it solved all instances available in the literature for the 2E-CVRP with up to 200 customers and 10 satellites, and besides, 34 instances were solved to optimality for the first time. To inspire further progress on solution approaches for the 2E-CVRP and related problems, we propose a new set of 51 instances to the community. Among them, 28 instances are currently open. Indeed, testing our algorithm on these new instances revealed that it has some limitations. We noticed that, for the largest instances, the size of the restricted master problem becomes so large that its resolution requires significant time. In a few extreme cases, a modern LP solver, such as CPLEX, cannot solve it within 1 hour. Moreover, the primal heuristic based on solving the restricted master is often inefficient because it cannot find improving primal solutions fast enough.

To overcome these difficulties, we believe that first-level routes should be generated dynamically. It is necessary if one wants to optimize instances with more than 15 satellites. Our new formulation for the problem without freight flow variables simplifies this task. However, column generation of first-level routes is not straightforward even for the modified formulation. A first-level route has coefficient one in constraints (2.12) if and only if it visits at least one satellite in a certain set. Thus, these constraints resemble strong capacity constraints introduced in Baldacci et al. (2008b). They are non-robust Pessoa et al. (2008), i.e. they modify the structure of the pricing problem for the first level.

In Chapter 3, we proposed an exact approach for the two-echelon capacitated vehicle routing problem with time windows in which freight storage and consolidation are allowed at satellites. Our algorithm tackles two variants of the problem: when city freighters do a single trip from a single satellite and when city freighters can do multiple trips visiting one or several satellites. Our problem variant is a relaxation of the variant considered in the literature. In our variant, we ensured precedence between urban trucks and city freighters at satellites whereas variants of the literature consider an exact synchronization of the vehicles at satellites. Similarly to Chapter 22, our approach is a branch-and-cut-and-price that includes techniques proposed recently for the classic vehicle routing problems. Since the problem is an extension of the 2E-CVRP, we use components that have proven to be efficient such as satellites supply inequalities and the branching strategy on the number of urban trucks visiting each satellite. Our algorithm also uses new problem-specific contributions such as an original route-based formulation with an exponential number of variables and constraints, the separation procedure for constraints linking the two levels, and a modification of the pricing problem structure to handle these constraints. Moreover, our algorithm adds dynamically first-level routes to the restricted master. It partially overcomes the main limitation of our algorithm for the 2E-CVRP.

We showed that our algorithm is efficient for the single-trip literature instances and some multi-trip literature instances with 100 customers. It outperforms significantly the only existing exact algorithm for the single-trip variant of the problem. It is also the first exact algorithm for the multi-trip variant of the problem. Moreover, we experimentally showed that our relaxation of the exact synchronization variant is very tight on the instances of the literature.

We propose new single-trip instances that are more difficult than the literature ones. The first set of instances is derived from instances of Solomon. The second set contains instances involving urban trucks whose capacity is not multiple of city freighter

capacity. Experimentations show that the latter property has a large impact on freight consolidation in the single-trip case.

The first research perspective is to improve the efficiency of our algorithm. Currently, the main limitation of our algorithm is due to the discretization of vehicle capacity in the pricing problem to take into account constraints linking the two distribution levels. Thus, instances with large city freighter capacity cannot be solved efficiently or sometimes even cannot be fit into memory. To overcome this limitation, we need to modify the labelling algorithm that solves the pricing problem. This algorithm should be able to work with arcs, for which the resource consumption is variable and the reduced cost depends on this consumption. An approach by Ioachim et al. (1998) can be useful here. Moreover, similarly to the algorithm for the 2E-CVRP, our algorithm relies on the enumeration of first-level routes. This enumeration is possible due to a small number of distribution centers and satellites in all instances of the literature. However, if this number gets larger, our approach will fail. To overcome this limitation, first-level routes should be generated by a pricing oracle similarly to the second-level routes. We will face the same problems as those previously described for the 2E-CVRP.

In Chapter 4, we proposed a branch-cut-and-price (BCP) algorithm for the location routing problem (LRP) and some related problems with the nested knapsack structure. Our main contribution consists in proposing new families of valid inequalities and separation algorithms for them. We used four families (COV, VCI, DCC, and FCC) of robust cuts. Only DCC was used before for the LRP. We proposed a family of Route Load Knapsack Cuts (RLKC) that contain non-robust cuts expressed over variables of the route-based formulation. Thus, we modified the pricing labelling algorithm which solves the resource constrained shortest path problem (RCSPP) to take into account the contribution of RLKCs to the reduced costs of paths. One positive result is that the necessary modifications do not have a large impact on the efficiency of the labelling algorithm.

We present the results of the computational evaluation of our BCP algorithm on literature instances of the LRP, on newly proposed instances for the related capacitated vehicle routing problem with multiple capacitated depots (CVRP-CMD), and on literature instances of the related vehicle routing problem with time windows and shifts (VRPTW-S). The results show that new families of cuts have a positive effect on the efficiency of the BCP algorithm. FCCs and VCIs contribute a lot to the success of the BCP algorithm for the LRP. RLKCs make difference for the CVRP-CMD and VRPTW-S. Finally, our BCP algorithm outperforms the current state-of-the-art approach (Contardo et al., 2014)

for the LRP, as it allows us to solve to optimality numerous test instances for the first time. As our algorithm is not based on the enumeration of subsets of open depots, it can be used for instances with a large number of potential depot locations.

Our experimental results show that the CVRP-CMD and the VRPTW-S remain difficult to solve, as our BCP algorithm cannot solve to optimality many instances with 50 customers. The main reason seems to be the quality of the linear relaxation which is not good enough, even if the extended formulation and all families of cuts are used. One possibility to improve the current results is to improve the separation algorithms for the cuts proposed in this work. Moreover, experimental results reveal that the quality of recent heuristics such as Schneider and Löffler (2019) is very good for small and medium LRP instances. There is still room for improvement for larger instances.

About the research perspectives, one could think of extending our algorithm for 2E-CVRP with time-windows and precedence constraints to the variant with exact synchronization of the two distribution levels. One could also think of extending our algorithm for 2E-CVRP to the two-echelon location-routing problem (Contardo et al., 2012) in which satellites have predefined capacities and fixed opening costs. Our preliminary research showed that additional valid inequalities are necessary for this variant of the problem. We ran out of time to try some components of chapter 4, such as FCC, on this problem.

Moreover, in this thesis, our effort was mainly to obtain tight valid lower bounds for the problems, and not to obtain feasible solutions. Another perspective could be to focus on the development of efficient heuristics for two-echelon vehicle routing problems. It seems promising to develop matheuristics, which are based on column generation or on branch-cut-and-price, for these problems.

Another important research direction is to be able to generate first-level routes dynamically. It is essential if one wants to optimize larger instances or try to plan routes for a distribution system with three or more levels. Therefore, we should carefully analyze the two-level balancing or precedence constraints and study how the labelling algorithm should be modified to take into account the contribution of these constraints to the reduced cost of first-level routes.

At last, one could think of evaluating the impact of such an optimization tool on real two-level distribution systems. This would also allow us to identify the limitations of our models and whether our assumptions are realistic.

Concerning the software perspectives, we saw that the exact approaches used in this thesis combine a lot of algorithms. Some algorithms are generic to integer optimization, others are dedicated to specific families of problems. During my thesis, I was involved in the development of Coluna which is a framework, written in Julia, to implement optimization methodologies based on decomposition and dynamic reformulation of mixed-integer linear programs. Basically, the user writes the original mixed integer program that models his problem using JuMP together with BlockDecomposition, which is a package that extends JuMP to specify the problem decomposition. Coluna automatically reformulates the original program following the decomposition instructions. Coluna then calls the algorithm chosen by the user to optimize the reformulation. Coluna has been designed to let the user build and run his own algorithm on the reformulation. Indeed, Coluna provides some algorithms, such that column generation or cut generation, and callbacks that the user can assemble to create his own algorithmic strategy. Preliminary experiments on the Generalized Assignement Problem are very encouraging. In the future, I plan to continue to contribute to the development of Coluna because I think that the Julia language is major progress to easily develop efficient applications and software for optimization.

Bibliography

Agence d'urbanisme Bordeaux Aquitaine (2019). La transport de marchandises en ville au sein de l'agglomération bordelaise. Technical report.

Akca, Z., Berger, R. T., and Ralphs, T. K. (2009). A branch-and-price algorithm for combined location and routing problems under capacity restrictions. In Chinneck, J. W., Kristjansson, B., and Saltzman, M. J., editors, *Operations Research and Cyber-Infrastructure*, pages 309–330, Boston, MA. Springer US.

Albareda-Sambola, M., Fernández, E., and Laporte, G. (2009). The capacity and distance constrained plant location problem. *Computers & Operations Research*, 36(2):597 – 611.

Allen, J., Browne, M., Woodburn, A., and Leonardi, J. (2012). The Role of Urban Consolidation Centres in Sustainable Freight Transport. *Transport Reviews*, 32(4):473–490.

Amarouche, Y., Guibadj, R. N., and Moukrim, A. (2018). A Neighborhood Search and Set Cover Hybrid Heuristic for the Two-Echelon Vehicle Routing Problem. In Borndörfer, R. and Storandt, S., editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, volume 65 of *OpenAccess Series in Informatics (OASIcs)*, pages 11:1–11:15, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Aráoz, J. (1974). *Polyhedral neopolarities*. PhD thesis, University of Waterloo, Department of Computer Science.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008a). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385.

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

Baldacci, R. and Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347–380.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011a). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research*, 61(2):298–314.

Baldacci, R., Mingozzi, A., and Wolfler Calvo, R. (2011b). An exact method for the capacitated location-routing problem. *Operations Research*, 59(5):1284–1296.

Belenguer, J.-M., Benavent, E., Prins, C., Prodhon, C., and Calvo, R. W. (2011). A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, 38(6):931 – 941.

Belenguer, J. M., Martinez, M. C., and Mota, E. (2000). A lower bound for the split delivery vehicle routing problem. *Operations Research*, 48(5):801–810.

Ben Mohamed, I., Klibi, W., Sadykov, R., Şen, H., and Vanderbeck, F. (2019). A Benders decomposition approach for the two-echelon stochastic multi-period capacitated location-routing problem. HAL 02178459, Inria.

Berger, R. T., Coullard, C. R., and Daskin, M. S. (2007). Location-routing problems with distance constraints. *Transportation Science*, 41(1):29–43.

Boccia, M., Sforza, A., Sterle, C., and Vasilyev, I. (2008). A cut and branch approach for the capacitated p-median problem based on fenchel cutting planes. *Journal of mathematical modelling and algorithms*, 7(1):43–58.

Boyd, E. A. (1993). Generating fenchel cutting planes for knapsack polyhedra. *SIAM Journal on Optimization*, 3(4):734–750.

Boyd, E. A. (1994). Fenchel cutting planes for integer programs. *Operations Research*, 42(1):53–64.

Breunig, U., Schmid, V., Hartl, R., and Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. *Computers and Operations Research*, 76:208 – 225.

Bulhoes, T., Sadykov, R., and Uchoa, E. (2018). A branch-and-price algorithm for the minimum latency problem. *Computers & Operations Research*, 93:66–78.

Chopra, S., Shim, S., and Steffy, D. E. (2015). A few strong knapsack facets. In Defourny, B. and Terlaky, T., editors, *Modeling and Optimization: Theory and Applications*, pages 77–94, Cham. Springer International Publishing.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2013a). A computational comparison of flow formulations for the capacitated location-routing problem. *Discrete Optimization*, 10(4):263 – 295.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, 26(1):88–102.

Contardo, C., Cordeau, J.-F. C., and Gendron, B. (2013b). A computational comparison of flow formulations for the capacitated location-routing problem. *Discrete Optimization*, 10(4):263 – 295.

Contardo, C., Hemmelmayr, V., and Crainic, T. G. (2012). Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & Operations Research*, 39(12):3185 – 3199.

Contardo, C. and Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129 – 146.

Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985.

Crainic, T. G. (2008). City Logistics. In *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*, INFORMS Tutorials in Operations Research, pages 181–212. INFORMS.

Crainic, T. G., Perboli, G., Mancini, S., and Tadei, R. (2010). Two-echelon vehicle routing problem: A satellite location analysis. *Procedia - Social and Behavioral Sciences*, 2(3):5944 – 5955.

Crainic, T. G., Ricciardi, N., and Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation science*, 43(4):432–454.

Crainic, T. G., Sforza, A., and Sterle, C. (2011). Location-routing models for two-echelon freight distribution system design. Technical Report 2011-40, CIRRELT.

Cuda, R., Guastaroba, G., and Speranza, M. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185 – 199.

Dabia, S., Ropke, S., and Van Woensel, T. (2019). Cover inequalities for a vehicle routing problem with time windows and shifts. *Transportation Science*, 53(5):1354–1371.

Dantzig, G. B. (1990). Origins of the simplex method. In *A history of scientific computing*, pages 141–151.

Dellaert, N., Saridarq, F. D., Woensel, T. V., and Crainic, T. G. (2019). Branch and price based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, In Press.

Dunning, I., Huchette, J., and Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320.

Fazel-Zarandi, M. M. and Beck, J. C. (2012). Using logic-based benders decomposition to solve the capacity- and distance-constrained plant location problem. *INFORMS Journal on Computing*, 24(3):387–398.

Gonzalez-Feliu, J., Perboli, G., Tadei, R., and Vigo, D. (2007). The two-echelon capacitated vehicle routing problem. Technical Report DEIS OR.INGCE 2007/2, Department of Electronics, Computer Science, and Systems, University of Bologna,.

Grangier, P., Gendreau, M., Lehuédé, F., and Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1):80–91.

Grötschel, M. and Nemhauser, G. L. (2008). George dantzig's contributions to integer programming. *Discrete Optimization*, 5(2):168–173.

He, P. and Li, J. (2019). The two-echelon multi-trip vehicle routing problem with dynamic satellites for crop harvesting and transportation. *Applied Soft Computing*, 77:387–398.

Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012a). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215 – 3228.

Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012b). An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228.

Holguín-Veras, J., Amaya Leal, J., Sanchez-Diaz, I., Browne, M., and Wojtowicz, J. (2018). State of the art and practice of urban freight management Part II: Financial approaches, logistics, and demand management. *Transportation Research Part A: Policy and Practice*.

Ibaraki, T. and Nakamura, Y. (1994). A dynamic programming method for single machine scheduling. *European Journal of Operational Research*, 76(1):72–82.

Ioachim, I., Gélinas, S., Soumis, F., and Desrosiers, J. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3):193–204.

Irnich, S., Desaulniers, G., Desrosiers, J., and Hadjar, A. (2010). Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, 22(2):297–313.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

Jepsen, M., Spoorendonk, S., and Ropke, S. (2013). A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, 47(1):23–37.

Kantorovich, L. V. (1939). The mathematical method of production planning and organization. *Management Science*, 6(4):363–422.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311.

Land, A. and Doig, A. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520.

Laporte, G. and Nobert, Y. (1981). An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6(2):224 – 226. Location Decisions.

Laporte, G. and Nobert, Y. (1983). A branch and bound algorithm for the capacitated vehicle routing problem. *Operations-Research-Spektrum*, 5(2):77–85.

Laporte, G., Nobert, Y., and Arpin, D. (1986). An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6(9):291–310.

Laporte, G., Nobert, Y., and Taillefer, S. (1988). Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science*, 22(3):161–172.

Li, H., Liu, Y., Chen, K., and Lin, Q. (2020a). The two-echelon city logistics system with on-street satellites. *Computers & Industrial Engineering*, 139:105577.

Li, H., Wang, H., Chen, J., and Bai, M. (2020b). Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B: Methodol*, 138:179–201.

Li, H., Zhang, L., Lv, T., and Chang, X. (2016). The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems. *Transportation Research Part B: Methodological*, 94:169–188.

Liguori, P. H. (2019). Polyhedral approaches for some network design problems.

Lysgaard, J. (2018). CVRPSEP: a package of separation routines for the capacitated vehicle routing problem.

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(2):423–445.

Marques, G., Sadykov, R., Deschamps, J.-C., and Dupas, R. (2020). An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Computers & Operations Research*, 114:104833.

McDermott, D. R. (1975). An Alternative Framework for Urban Goods Distribution: Consolidation. *Transportation Journal*, 15(1):29–39.

Muter, I., Cordeau, J.-F., and Laporte, G. (2014). A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, 48(3):425–441.

Nolz, P. C., Absi, N., Cattaruzza, D., and Feillet, D. (2020). Two-echelon distribution with a single capacitated city hub. *EURO Journal on Transportation and Logistics*, page 100015.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017a). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., and Santos, H. (2017b). Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, 45(3):206 – 209.

Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380.

Pessoa, A., de Aragão, Marcus, M. P., and Uchoa, E. (2008). Robust branch-cut-and-price algorithms for vehicle routing problems. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 297–325. Springer US.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2018). Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming B*, accepted.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483.

Prins, C., Prodhon, C., and Wolfler Calvo, R. (2006). Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR*, 4(3):221–238.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273.

Roberti, R. and Mingozzi, A. (2014). Dynamic ng-path relaxation for the delivery man problem. *Transportation Science*, 48(3):413–424.

Rothberg, E. (2007). An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541.

Sadykov, R., Uchoa, E., and Pessoa, A. (2020). A bucket graph based labeling algorithm with application to vehicle routing. *Transportation Science*, accepted.

Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., and Uchoa, E. (2018). Primal heuristics for branch-and-price: the assets of diving methods. *INFORMS Journal on Computing*, In Press.

Salhi, S. and Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150 – 156.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2015). A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Transportation Science*, 49(2):355–368.

Schneider, M. and Drexl, M. (2017). A survey of the standard location-routing problem. *Annals of Operations Research*, 259(1):389–414.

Schneider, M. and Löffler, M. (2019). Large composite neighborhoods for the capacitated location-routing problem. *Transportation Science*, 53(1):301–318.

Schneider, M. and Löffler, M. (2019). Large composite neighborhoods for the capacitated location-routing problem. *Transportation Science*, 53(1):301–318.

Taniguchi, E. and Thompson, R. G. (2002). Modeling city logistics. *Transportation research record*, 1790(1):45–51.

Tilk, C., Olkis, K., and Irnich, S. (2020). The last-mile vehicle routing problem with delivery options. discussion paper 2017, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Tuzun, D. and Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European journal of operational research*, 116(1):87–99.

Vanderbeck, F. (2005). Implementing mixed integer column generation. In *Column generation*, pages 331–358. Springer.

Vanderbeck, F., Sadykov, R., and Tahiri, I. (2019). BaPCod — a generic Branch-And-Price Code.

Wang, K., Shao, Y., and Zhou, W. (2017). Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. *Transportation Research Part D: Transport and Environment*, 57:262 – 276.

Wang, Z. and Wen, P. (2020). Optimization of a low-carbon two-echelon heterogeneous-fleet vehicle routing for cold chain logistics under mixed time window. *Sustainability*, 12(5):179–201.

Wentges, P. (1997). Weighted dantzig-wolfe decomposition for linear mixed-integer programming. *International Transactions in Operational Research*, 4(2):151–162.

Wolsey, L. A. (1998). *Integer programming*. John Wiley & Sons.

Zeng, Z.-Y., Xu, W.-S., Xu, Z.-Y., and Shao, W.-H. (2014). A hybrid GRASP+VND heuristic for the two-echelon vehicle routing problem arising in city logistics,. *Mathematical Problems in Engineering*, pages 1–11.