Université de Montréal

Models and Algorithms for the Capacitated Location-Routing Problem

par
Claudio Contardo

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

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Juillet 2011

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

Models and Algorithms for the Capacitated Location-Routing Problem

présentée par:

Claudio Contardo

a été évaluée par un jury composé des personnes suivantes:

Fabian Bastin,	président-rapporteur
Bernard Gendron,	directeur de recherche
Jean-François Cordeau,	codirecteur
Gilbert Laporte,	membre du jury
Roberto Wolfler-Calvo,	examinateur externe
Claude Comtois,	représentant du doyen de la FAS

Thèse acceptée le: . . . . . .

RÉSUMÉ

Le problème de localisation-routage avec capacités (PLRC) apparaît comme un problème clé dans la conception de réseaux de distribution de marchandises. Il généralise le problème de localisation avec capacités (PLC) ainsi que le problème de tournées de véhicules à multiples dépôts (PTVMD), le premier en ajoutant des décisions liées au routage et le deuxième en ajoutant des décisions liées à la localisation des dépôts. Dans cette thèse on dévelope des outils pour résoudre le PLRC à l'aide de la programmation mathématique. Dans le chapitre 3, on introduit trois nouveaux modèles pour le PLRC basés sur des flots de véhicules et des flots de commodités, et on montre comment ceux-ci dominent, en termes de la qualité de la borne inférieure, la formulation originale à deux indices [19]. Des nouvelles inégalités valides ont été dévelopées et ajoutées aux modèles, de même que des inégalités connues. De nouveaux algorithmes de séparation ont aussi été dévelopés qui dans la plupart de cas généralisent ceux trouvés dans la litterature. Les résultats numériques montrent que ces modèles de flot sont en fait utiles pour résoudre des instances de petite à moyenne taille. Dans le chapitre 4, on présente une nouvelle méthode de génération de colonnes basée sur une formulation de partition d'ensemble. Le sous-problème consiste en un problème de plus court chemin avec capacités (PCCC). En particulier, on utilise une relaxation de ce problème dans laquelle il est possible de produire des routes avec des cycles de longueur trois ou plus. Ceci est complété par des nouvelles coupes qui permettent de réduire encore davantage le saut d'intégralité en même temps que de défavoriser l'apparition de cycles dans les routes. Ces résultats suggèrent que cette méthode fournit la meilleure méthode exacte pour le PLRC. Dans le chapitre 5, on introduit une nouvelle méthode heuristique pour le PLRC. Premièrement, on démarre une méthode randomisée de type GRASP pour trouver un premier ensemble de solutions de bonne qualité. Les solutions de cet ensemble sont alors combinées de façon à les améliorer. Finalement, on démarre une méthode de type *détruir et réparer* basée sur la résolution d'un nouveau modèle de localisation et réaffectation qui généralise le problème de réaffectaction [48].

Mots clés: localisation-routage, géneration de colonnes, heuristiques.

ABSTRACT

The capacitated location-routing problem (CLRP) arises as a key problem in the design of distribution networks. It generalizes both the capacitated facility location problem (CFLP) and the multiple depot vehicle routing problem (MDVRP), the first by considering additional routing decisions and the second by adding the location decision variables. In this thesis we use different mathematical programming tools to develop and specialize new models and algorithms for solving the CLRP. In Chapter 3, three new models are presented for the CLRP based on vehicle-flow and commodity-flow formulations, all of which are shown to dominate, in terms of the linear relaxation lower bound, the original two-index vehicle-flow formulation [19]. Known valid inequalities are complemented with some new ones and included using separation algorithms that in many cases generalize extisting ones found in the literature. Computational experiments suggest that flow models can be efficient for dealing with small or medium size instances of the CLRP (50 customers or less). In Chapter 4, a new branch-and-cut-and-price exact algorithm is introduced for the CLRP based on a set-partitioning formulation. The pricing problem is a shortest path problem with resource constraints (SPPRC). In particular, we consider a relaxation of such problem in which routes are allowed to contain cycles of length three or more. This is complemented with the development of new valid inequalities that are shown to be effective for closing the optimality gap as well as to restrict the appearance of cycles. Computational experience supports the fact that this method is now the best exact method for the CLRP. In Chapter 5, we introduce a new meta-heuristic with the aim of finding good quality solutions in short or moderate computing times. First, a bundle of good solutions is generated with the help of a greedy randomized adaptive search procedure (GRASP). Following this, a blending procedure is applied with the aim of producing a better upper bound as a combination of all the others in the bundle. An iterative destroy-and-repair method is then applied using a location-reallocation model that generalizes the reallocation model due to de Franceschi et al. [48].

Keywords: location-routing, branch-and-cut, branch-and-price, metaheuristic.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

Algorithms

2-cyc-SPPRC	Shortest Path Problem With Resource Constraints and 2-Cycle Elimination
ALNS	Adaptive Large Neighborhood Search
CWSA	Clarke and Wright Savings Algorithm
ECWSA	Extended Clarke and Wright Savings Algorithm
ESPPRC	Elementary Shortest Path Problem With Resource Constraints
GRASP	Greedy Randomized Adaptive Search Procedure
LIH	Local Improvement Heuristic
LRGTS	Cooperative Lagrangean Relaxation-Granular Tabu Search
LS	Local Search
MA	Memetic Algorithm
MAPM	Memetic Algorithm with Population Management
RECWSA	Randomized Extended Clarke and Wright Savings Algorithm
SA	Simulated Annealing
SB	Solution Blender
SPPRC	Shortest Path Problem With Resource Constraints
TS	Tabu Search
VNS	Variable Neighborhood Search

Combinatorial Optimization Problems

BPP	Bin-Packing Problem
CFLP	Capacitated Facility Location Problem
CLRP	Capacitated Location-Routing Problem
CVRP	Capacitated Vehicle Routing Problem
FLP	Facility Location Problem
LRP	Location-Routing Problem
MDVRP	Multiple Depot Vehicle Routing Problem
MSFLP	Multiple-Source Facility Location Problem
SPLP	Simple Plant Location Problem
SSFLP	Single-Source Facility Location Problem
TSP	Traveling Salesman Problem
UFLP	Uncapacitated Facility Location Problem
ULRP	Uncapacitated Location-Routing Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem With Time Windows

Mathematical Programming

CC	Capacity Cuts
CFF	Commodity-Flow Formulation
CoCC	Co-Circuit Constraints
DCoCC	Disaggregated Co-Circuit Constraints
ESFCI	Effective Strengthened Facility Capacity Inequalities
FAI	Flow-Assignment Inequalities
FCI	Facility Capacity Inequalities
FDC	Facility Degree Constraints
FrCI	Framed Capacity Inequalities
GLM	Generalized Large Multistar Inequalities
HYP	Hypotour Inequalities
ILP	Integer-Linear Problem
LCI	Lifted Cover Inequalities
LRCOMB	Location-Routing Comb Inequalities
LRGLM	Location-Routing Generalized Large Multistar Inequalities
LRM	Location-Reallocation Model
MSI	Multistar Inequalities
RM	Reallocation Model
SCI	Strengthened Comb Inequalities
SDEG	Strengthened Degree Constraints
SFCI	Strengthened Facility Capacity Inequalities
SFrCI	Strengthened Framed Capacity Inequalities
SPC	Strengthened Path Constraints
SPF	Set-Partitioning Formulation
SP-ESFCI	Set-Partitioning Effective Strengthened Facility Capacity Inequalities
SP-SFCI	Set-Partitioning Strengthened Facility Capacity Inequalities
SRI	Subset-Row Inequalities

Mathematical Programming

VFF	Vehicle-Flow Formulation
y-CC	y-Capacity Cuts
y-GLM	y-Generalized Large Multistar Inequalities
y-LRGLM	y-Location-Routing Generalized Large Multistar Inequalities
y-SCC	y-Strengthened Capacity Cuts

Miscellaneous

MTZ	Miller-Tucker-Zemlin
$\mathscr{NP}$	Non-deterministic polynomial time

To the love of my life and future wife, Clara.

ACKNOWLEDGMENTS

I would like to start thanking my advisors, Professors Jean-François Cordeau and Bernard Gendron, who trusted in me since the very beginning. Without their continuous support and fruitful discussions, this dissertation would not have been possible. I am deeply thankful of the many hours that you spent teaching and guiding me.

I would like to thank my best friend, lover, accomplice, lawyer and nurse (among many other tasks), Clara. Since the day we met you have been the light in my heart, the sunshine and warmth that may not come very often at certain times of the year, but with which I wake up every day since two and a half years now.

I would like to thank my family in Chile, my parents, brothers, aunts, uncles, grandparents, cousins and nephews. You were my spiritual support and source of infinite and unconditional love. These -somethimes short, sometimes long- conversations on the chat, on the phone as well as the the time spent when I was back in my country, were my charge of battery.

I would like to thank my friends in Montréal, continuous sources of support, laugh and distractions. Always holding a glass in our hands you made of my stay in Montréal an unforgettable experience: Gerardo Berbeglia, John Alexander López, Angely Jamis, Aymen Karoui, Malek Ben-Jemia, Daniel Aloise, Caroline Rocha, Alysson Costa, Iván Contreras, Mayret Talonia. There are plenty of other people, but if I start writing on their names I would need another 200 pages.

Last but not least, I would like to thank the *Natural Sciences and Engineering Research Council of Canada* (NSERC) and *Le Fonds québécois de la recherche sur la nature et les technologies* (FQRNT), for their financial support, without which I would have starved to death, and this thesis would not have been possible.

CHAPTER 1

INTRODUCTION

Combinatorial optimization is an important field in the area of computer science and operations research. Many industrial applications, such as the location of warehouses, the routing of vehicles in freight distribution, or the scheduling of employees in a supermarket, can be modeled as combinatorial optimization problems. Some of these problems, due to their specific structure, are known to be easily solvable. That is the case of shortest path problems, minimum spanning tree problems or sorting problems, for which polynomial-time exact algorithms are known. However, many other problems fall into the category of $\mathscr{NP}$-hard problems, for which no polynomial-time algorithms are known. For a comprehensive formalization of these concepts, the reader is referred to some classic literature in combinatorial optimization and complexity theory [60, 114]. However, it is worth mentioning that for those problems known to be $\mathscr{NP}$-hard, polynomial-time algorithms are unlikely to exist unless $\mathscr{P} = \mathscr{NP}$.

The fact that no polynomial-time algorithm is known for a certain problem does not mean that no *efficient* algorithms are known for it. Indeed, some problems, although $\mathscr{NP}$-hard, present some nice structures that can be exploited and then efficiently solved either exactly or heuristically. That is the case, for instance, of the 0-1 knapsack problem which is known to be *weakly* $\mathscr{NP}$-hard and that can be solved exactly in pseudo-polynomial time using dynamic programming. Problems that are not *weakly* $\mathscr{NP}$-hard are also referred to as *strongly* $\mathscr{NP}$-hard. A typical case of a *strongly* $\mathscr{NP}$-hard problem is the traveling salesman problem, for which efficient algorithms are able to deal with very large instances.

The main objective of this thesis is to introduce models and efficient algorithms for the capacitated location-routing problem (CLRP), a rich combinatorial optimization problem arising in many real-life applications, such as the location of warehouses and the distribution of commodities from those warehouses to customers. In the CLRP, a decision maker must decide of the location of facilities and the routing of vehicles in

order to satisfy the known demand of a set of customers. This problem is a generalization of two combinatorial optimization problems: the capacitated facility location problem (CFLP) and the capacitated vehicle routing problem (CVRP), both of which are known to be *strongly* $\mathscr{NP}$-hard and for which efficient exact and heuristic agorithms exist. Because of this, the CLRP also belongs to the class of *strongly* $\mathscr{NP}$-hard problems.

To achieve this objective, this thesis is divided into three chapters that present the contributions made to the CLRP. In Chapter 3, we introduce three new formulations of the CLRP based on vehicle flows and commodity flows, which are proven to dominate, in terms of the linear relaxation lower bound, the original two-index vehicle-flow formulation of Belenguer et al. [19] at the expense of adding more variables. We derive two new families of multistar inequalities from the commodity-flow formulations and introduce separation algorithms for using them inside the vehicle-flow formulations. We also introduce several new families of valid inequalities for the formulations introduced, and strengthen several of the existing ones, which are complemented with new, efficient separation algorithms, which in many cases generalize those introduced by Belenguer et al. [19]. We perform a computational study comparing each of the formulations on a large number of instances and discuss the strengths and weaknesses of each formulation. The results obtained show that the compact two-index vehicle-flow formulation is in general more robust than the others. Indeed, the slightly worse lower bounds achieved with this formulation are usually compensated by the larger number of branching nodes that can be explored within a branch-and-cut algorithm, leading to better average optimality gaps. Additionally, our implementation of the branch-and-cut algorithm over the two-index vehicle-flow formulation scales better than the one introduced by Belenguer et al. [19], being able to solve instances with up to 100 customers whereas the original method of Belenguer et al. [19] was only capable of solving instances containing up to 50 customers.

In Chapter 4, we present a branch-and-cut-and-price algorithm for the CLRP. We adapt the set-partitioning formulation of Akca et al. [4] so that all of the cuts valid for the previous formulations can be incorporated. We introduce two bounding procedures that are applied sequentially and that allow, in most cases, to reduce the CLRP to a series

of MDVRPs. Our computational results show that our bounding procedures can in fact be stronger than those of Baldacci et al. [16] in some instances. We also introduce several new families of cuts that are effective for closing the optimality gap. One of the families introduced allows the use state-space relaxation in the pricing problem so as to get lower bounds close to those obtained if pricing on elementary routes (routes that do not contain cycles). This is complemented with new fathoming rules that accelerate the solution of the pricing subproblems. The results obtained on several sets of instances show that our method obtains better lower bounds than that of Baldacci et al. [16] and is able to solve to optimality several open instances, and to improve the best known solutions for some others.

In Chapter 5, we present a new heuristic algorithm for the CLRP based on the sequential application of a GRASP metaheuristic and the solution of several integer-linear programs (ILP). Our implementation of the GRASP algorithm yields better average results than the previous approach of Prins et al. [121]. We also introduce a novel location-reallocation model (LRM) that takes into account the location and the routing decisions simultaneously. The proposed model is based on a set-partitioning formulation that generalizes both the CFLP and the reallocation model of de Franceschi et al. [48], the first by adding the possibility of inserting customers in the middle of the routes, and the second by adding the possibility of reallocating whole routes to different facilities. We introduce a new technique based on the solution of the LRM for combining a bundle of reasonably good solutions with the objective of producing an improved solution. Our computational experience shows that our algorithm provides better average results than previous methods in competitive times and is able to improve the best known feasible solutions on several instances from the literature.

CHAPTER 2

LITERATURE REVIEW

In this chapter we provide an extensive literature review of the research concerning the different classes of location-routing problems (LRP) and of closely related problems such as facility location problems (FLP) and vehicle routing problems (VRP), including some particular cases of these. We first give a brief description of the different classes of FLP, VRP and LRP that can be found in the literature and describe the different algorithmic alternatives that have been devised to deal with their variants. We then give a more detailed review of the CLRP including the different modeling approaches and existing algorithms.

2.1 An overview of location-routing and related problems

In this section we give a short survey of the location-routing problem and two closely related problems, the facility location problem and the vehicle routing problem. It includes a brief description of each of these problems including several particular cases, as well as the corresponding algorithms found in the literature.

2.1.1 Facility location problems

Facility location problems arise as important problems in many industrial applications, such as the location of bank accounts among the different branches of a bank and the positioning of mobile phone antennaes, just to name a few. In their more general form, given a set of potential facilities I and a set of customers J, they consist to find a subset of facilities $I' \subseteq I$ and to assign customers to those facilities so as to optimize some objective function. The set I might be of finite or infinite size, and also the objective function may take different forms. Recent surveys of different classes of facility location problems and algorithmic approaches can be found in Daskin [47], Revelle et al. [127] and Smith et al. [136]. The seminal works of Hakimi [73, 74] introduce the *p-median*

and *p-center* problems. In the *p-median* problem, given a set of customer locations J and a finite set of potential facility locations I, the problem is to select a subset of facilities I' of cardinality p and to assign customers to these open facilities so as to minimize the sum of the distances from the open facilities to their assigned customers. In the *p-center* problem, the input is the same as for the *p-median*, but the objective is now to select p facilities and assign customers to these facilities so as to minimize the maximum distance between any facility and its assigned customers. These two problems belong to the class of $\mathscr{NP}$-hard problems [81, 82]. Many algorithms, most of them heuristics, have been proposed for these problems. For the *p-median* problem, the classical references include the vertex substitution method of Teitz and Bart [139] and the decomposition method of Garfinkel et al. [61]. A recent survey by Mladenovic et al. [106] recapitulates the extensive literature for this problem. If the number of facilities to locate is not known in advance, but one instead considers their setup costs, the natural extension of the *p-median* problem is the so-called simple plant location problem (SPLP), introduced by Kuehn and Hamburger [85]. One of the most efficient exact algorithm for the SPLP is the dual-ascent method introduced by Erlenkotter [54]. The SPLP is closely related to some classical combinatorial optimizatiom problems like the set-covering problem or the set-partitioning problem. Polyhedral studies of the SPLP exploit these similarities in order to derive valid inequalities and prove conditions under which these inequalities induce facets of the SPLP polytope [28, 29, 41]. A natural extension of the SPLP is the case in which facilities have limited capacities. Such a problem is known as the capacitated facility location problem (CFLP). Depending on whether customers can be served by several facilities or just one, we distinguish between the multiple source FLP (MSFLP) or the single source FLP (SSFLP). For the MSFLP, Lagrangean methods seem to be a very promising avenue. Depending on which sets of constraints are relaxed one gets different approaches [see for example 30, 63, 143]. Several studies propose valid inequalities for the MSFLP polytope and devise conditions under which these inequalities induce facets [1, 2, 96]. Some of these inequalities come from related problems as the SPLP [28], fixed-charge network-design problems [145] and lot-sizing problems [118]. The usefulness of these inequalities is unclear as their separation algorithms in most

cases are $\mathscr{NP}$-hard. Regarding the SSFLP, decomposition methods have shown to be the most successful approaches. Neebe and Rao [111] were the first to formulate the SSFLP as a set-partitioning problem and solved it by branch-and-price. Holmberg et al. [79] developed a Lagrangean heuristic by relaxing the assignment constraints. Their algorithm, despite being old, remains one of the most efficient exact algorithms for the SSFLP. Other Lagrangean heuristics have been developed by Barceló and Casanovas [17] and Klincewicz and Luss [84]. Díaz and Fernández [52] developed a branch-and-price method in which the subproblem reduces to solving a series of 0-1 knapsack problems. Metaheuristics, such as the very large-scale neighborhood search by Ahuja et al. [3], have also been proposed to deal with large-size instances of the SSFLP.

2.1.2 Vehicle routing problems

Vehicle routing problems arise as important tactical/operational problems in many industrial applications, such as the distribution of mail, the schedule of school bus routes and the routing of maintenance units, just to name a few. The simplest VRP can be stated as follows. Given a set of customers J, the problem is to find the Hamiltonian cycle (a simple cycle containing all the nodes in J) so as to minimize the total length of that cycle. This problem is known in the literature as the traveling salesman problem (TSP), and was first introduced by Dantzig et al. [46]. This problem is also known to be $\mathscr{NP}$-hard [60]. The TSP has a very rich combinatorial structure and it is considered as one of the simplest problems presenting such practical interest and algorithmic challenges. For these reasons, many algorithms, both exact and heuristics, have been developed to solve the TSP. Several surveys and books cover some of the most important contributions made to this problem [5, 20, 72, 93, 126]. In the seminal work of Dantzig et al. [46] the TSP is formulated as a linear-integer program. The authors introduced the first two-index formulation for this problem and solved it by means of branch-and-cut, a novel idea at that time. The problem contains an exponential number of subtour elimination constraints (a subtour is a tour containing strictly less than $|J|$ nodes) that are first relaxed and then dynamically added to the problem. Their algorithm was able to solve an instance with 49 customers, one for each state of the continental United States. Miller et al. [105] intro-

duced the so-called MTZ constraints for the TSP, which replace the subtour elimination constraints of Dantzig et al. [46] by ensuring the elimination of subtours after adding a polynomial number of variables and constraints. This formulation, however, produces weaker lower bounds than the original formulation of Dantzig et al. [46]. Nowadays, the most succesful approaches to solving the TSP are branch-and-cut algorithms [34]. Several families of valid inequalities have been introduced [24, 33, 58, 69, 94] to strengthen the linear relaxation of this problem at the point of allowing the solution of some very large instances. The VRP arises as a natural extension of the TSP when vehicle capacities need to be explcitely taken into account. In the VRP, in addition to the customer locations J, an extra node is added to represent a facility. Then, the problem is to route a given fleet of capacitated vehicles, each of them leaving from and returning to the facility, in order to visit each customer exactly once and such that vehicle capacities are respected. The goal is to minimize the total traveling cost. The literature on the VRP is vast, and it includes several surveys and compendiums [39, 65, 88, 140]. The first to formally introduce this problem were Dantzig and Ramser [45]. Among the heuristic methods for solving the VRP, we first mention 1-phase or constructive methods [31, 35, 107] that build a solution by iteratively adding customers to a current partial solution. Clarke and Wright [35] introduced the concept of a *savings algorithm* in which, starting from a feasible solution, routes are merged using a descent criteria. Two-phase (cluster first, route second) methods [25, 57] separate the problem in two subproblems that are solved sequentially. Metaheursitics methods [see 62, among others] explore the solution space by jumping from one solution to another aiming to reach a global optima which otherwise might not be reachable by the 1-phase or 2-phase heuristics which usually converge to a local optimum. At each iteration a solution (sometimes unfeasible) is provided, and a neighborhood exploration is performed looking for a better solution close to it. This exploration may result in a deterioration of the total cost and this is how local optima is avoided. Pisinger and Røpke [117] introduced an adaptive large neighborhood search heuristic (ALNS) able to find very good quality solutions for large instances in a short time. The main idea of their algorithm is to combine the large neighborhood search approach of [134], in which a series of destroy and repair methods are performed se-

quentially in order to explore the neighborhood space, with an adaptive procedure that identifies the methods that perform the best in the destroy and repair process. Regarding the exact methods developed for solving the VRP to optimality, it is possible to mention the polyhedral studies on the two-index formulation of Dantzig and Ramser [45] for the particular case of symmetric costs and homogeneous fleet. This problem is better known as the capacitated VRP (CVRP), for which several authors have proposed valid inequalities and separation algorithms [6, 40, 42, 95, 108, 109]. Probably the most succesful exact method based on cutting planes for the CVRP is the one of Lysgaard et al. [101] which includes efficient separation algorithms for several classes of valid inequalities. However, large-size instances can be very hard to solve even for the most sophisticated branch-and-cut algorithms. Recently, column generation-based methods have been developed that scale much better than flow formulations at the expense of adding much more variables. These methods are based on strong set-partitioning formulations of the CVRP and provide very tight lower bounds. Among these approaches, it is possible to identify the works of Fukasawa et al. [59] and Baldacci et al. [14]. Another natural extension of the VRP is the multiple depot VRP (MDVRP), in which instead of considering just one facility, vehicles are routed from a set of facilities I. For each facility $i \in I$, one associates a capacity b_i representing the maximum amount of commodity that can be served from that facility. The objective is to route the vehicles from the different facilities at minimum cost so as to serve each customer exactly once while respecting both vehicle and facility capacities. This problem is usually much more difficult than the VRP and most of the literature has focused on the development of efficient heuristic algorithms. Cordeau et al. [38] developed a tabu search heuristic for the MDVRP. Contrary to descent algorithms, after each iteration the solution may deteriorate. A tabu list forbids, however, cycling between a good and a bad solution, and it also adds diversification to the search. Pisinger and Røpke [117] use an adaptation of their ALNS method to deal with the MDVRP. Vidal et al. [144] proposed a hybrid genetic algorithm for multiple classes of vehicle routing problems, including the MDVRP. Their algorithm is very robust and usually improves or at least finds the best known feasible solutions for the instances considered in their study. Baldacci and Mingozzi [12] introduced a general

framework to deal with several different classes of vehicle-routing problems, including the MDVRP. A set-partitioning formulation of the MDVRP is provided and solved by means of branch-and-cut-and-price.

2.1.3 Location-routing problems

The location-routing problem arises as a natural extension of both the FLP and the VRP. Many applications include interactions between location and routing decisions, such as the location of distribution centers or warehouses for multiple types of commodities from which goods are distributed to customers, or more recently some applications arising in city logistics [43]. By neglecting this interaction, it is possible to approximately solve this problem as a pure location problem. However, it is known that making such simplification may lead to sub-optimal solutions of very poor quality [131]. In its simplest form, the LRP can be stated as follows. Given a set of potential facilities I (that may be of infinite size), a set of customers J and a fleet of vehicles K, the problem is to select a subset of facilities I', to route the vehicles of set K from these facilities and to satisfy each customer's demand so as to minimize a certain objective function. Most of the literature on location-routing problems deal with the particular case in which I is of finite size. Only a few papers [130, 133] deal with the problem of locating a single facility in the contiuous space. Otherwise, the location-routing literature has mainly focused on the problem of locating an arbitrary number of facilities. Some studies [9, 10, 89, 133, 135] consider the particular case in which the size of I' is determined in advance. Laporte [87] and Nagy and Salhi [110] provide complete surveys of the models and algorithms for the LRP existing up to the date they were published. The paper of Laporte [87] introduces a general three-index formulation that includes as special cases those given by Golden et al. [66], Or and Pierskalla [112], Srikar and Srivastava [138] and Perl and Daskin [115]. Regarding the more general case in which the set I has finite size and the size of the set I' is to be determined, Laporte et al. [91] propose the first two-index vehicle-flow formulation of the LRP with uncapacitated facilities (ULRP) introducing chain barring constraints to eliminate solutions where vehicle routes start and finish at different facilities. Laporte et al. [92] propose a graph transformation of the ULRP to

reformulate it as TSP. They embed it into a branch-and-bound algorithm so that at each node in the branching tree the subproblem reduces to an assignment problem. Cappanera et al. [26] introduce a two-index commodity-flow formulation of the problem in the context of locating facilities for the routing of obnoxious materials. In their modeling approach, customers are allowed to be served by several vehicles. They solve the problem by means of a branch-and-bound algorithm in which lower bounds are obtained by Lagrangean relaxation. The Lagrangean subproblems consist in solving a 0-1 multidimensional knapsack problem and a routing problem. The capacitated LRP (CLRP) can be seen as a particular case of the LRP in which the vehicle fleet is homogeneous, of infinite size, the network is symmetric and facilities have limited capacities. Belenguer et al. [19] were the first to give a mathematical formulation of this problem, by extending the formulation of Laporte et al. [91] to consider facilities with limited capacities. The authors propose several families of valid inequalities and embed them into a branch-and-cut algorithm. Their algorithm succeeds to solve instances with up to 50 customers. Another exact algorithm, developed by Baldacci et al. [16], formulates the problem as a set-partitioning problem in which variables represent feasible routes with respect to vehicle capacities. A three-phase branch-and-cut-and-price method is applied which reduces the problem to a (usually) small number of MDVRP. Their algorithm is able to solve to optimality instances containing up to 199 customers. For dealing with larger instances of the CLRP, several heuristics have been introduced, namely GRASP methods [121], memetic algorithms [120], tabu search [122], simulated annealing [147], hybrid metaheuristics combining variable neighborhood search with integer linear programming methods [116] and adaptive large neighborhood search algorithms [78].

2.2 The capacitated location-routing problem

The capacitated location-routing problem (CLRP) is a particular case of the LRP and can be stated as follows. Given a finite set of potential facilities I, each facility having a setup cost f_i and a capacity b_i, a set of customers J, where each customer has a demand d_j, an homogeneous fleet of infinite size, where each vehicle has the same

capacity Q, and an underlying undirected graph $G = (V = I \cup J, E)$, with associated cost matrix $(c_e)_{e \in E}$, the goal is to select a subset of facilities $I' \subseteq I$ and to route vehicles from these facilities so as to visit each customer exactly once at minimum overall cost while respecting both facility and vehicle capacities. In the following subsections we describe the different mathematical formulations for this problem as well as the existing algorithms, including both exact and heuristic methods.

2.2.1 Mathematical formulations

In this subsection we describe two mathematical formulations for the CLRP. The first is a two-index vehicle-flow formulation introduced by Belenguer et al. [19]. The second is a set-partitioning formulation introduced by Akca et al. [4].

2.2.1.1 Two-index vehicle-flow formulation

Let $G = (V,E)$ be an undirected graph, where $V = I \cup J$ and $E = \{\{v_i, v_j\} : v_i, v_j \in V\} \setminus I \times I$. For every subset $U \subseteq V$, we define $E(U) = \{\{u,w\} \in E : u,w \in U\}$, and $\delta(U) = \{\{u,w\} \in E : u \in U, w \notin U\}$. For every pair of disjoint subsets U and W, let also $(U : W) = \{\{u,w\} \in E : u \in U, w \in W\}$. With every edge $e \in \delta(I)$ are associated two binary variables: x_e equal to 1 iff edge e is used once, and y_e equal to 1 iff edge e is used twice. With every edge $e \in E(J)$ is associated a binary variable x_e equal to 1 iff edge e is used. For every facility $i \in I$, let z_i be a binary variable equal to 1 iff facility i is selected. For a given edge set $F \subseteq E$ we define $x(F) = \sum_{e \in F} x_e$ and $y(F) = \sum_{e \in F} y_e$ (if $F \subseteq \delta(I)$). For a given subset $S \subseteq J$ of customers, we define $d(S) = \sum_{j \in S} d_j$, and a constant $r(S) = \lceil d(S)/Q \rceil$ which is a lower bound on the number of vehicles required to satisfy the demand of customers in S. Finally, we define $\overline{S} = J \setminus S$. The CLRP can then be formulated as the following integer program.

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{e \in E} c_e x_e + 2 \sum_{e \in \delta(I)} c_e y_e \tag{VF2}$$

subject to

$$x(\delta(j)) + 2y(I:\{j\}) = 2 \qquad j \in J \qquad (2.1)$$

$$x(\delta(S)) + 2y(I:S) \geq 2r(S) \qquad S \subseteq J, |S| \geq 2 \qquad (2.2)$$

$$x_{ij} + y_{ij} \leq z_i \qquad i \in I, j \in J \qquad (2.3)$$

$$x(I:\{j\}) + y(I:\{j\}) \leq 1 \qquad j \in J \qquad (2.4)$$

$$x((I \setminus \{i\}) \cup \overline{S}:S) + 2y(I \setminus \{i\}:S) \geq 2 \qquad i \in I, S \subseteq J, d(S) \geq b_i \qquad (2.5)$$

$$x(\delta(S)) \geq 2(x(\{h\}:I') + x(\{j\}:I \setminus I')) \qquad S \subseteq J, |S| \geq 2, h, j \in S, I' \subset I \qquad (2.6)$$

$$z_i \in \{0,1\} \qquad i \in I \qquad (2.7)$$

$$x_e \in \{0,1\} \qquad e \in E \qquad (2.8)$$

$$y_e \in \{0,1\} \qquad e \in \delta(I). \qquad (2.9)$$

Demand constraints (2.1) impose that every customer vertex be visited once and also act as flow conservation equations. Constraints (2.2) are the capacity cuts (CC) which play a dual role: they forbid tours disconnected from facilities as well as tours serving a demand larger than Q. Constraints (2.3) ensure that there is no outgoing flow from unselected facilities. Constraints (2.4) forbid single-customer routes to be linked to two different facilities. Constraints (2.5) are the facility capacity inequalities (FCI). They forbid the existence of a set of routes leaving from a given facility i and serving a demand higher than b_i. Constraints (2.6) are the path constraints (PC) that prevent the route of a vehicle from joining two different facilities. These constraints are not valid when $|S| = 1$ and they are thus complementary to constraints (2.4).

Unlike in traditional CVRP formulations, two sets of variables (x and y) are associated with the edges in $\delta(I)$. One can in fact check that if these variables are replaced with the aggregated variables $\bar{x}_e = x_e + 2y_e$, single-customer routes linked to two different facilities can no longer be correctly eliminated as we do with constraints (2.4).

2.2.1.2 Set-partitioning formulation

Let us denote by Ω_i the set of all routes (possibly containing cycles) starting and ending at facility $i \in I$ and servicing a subset of customers with a total demand of Q or less, and let $\Omega = \cup_{i \in I} \Omega_i$ be the set of all possible routes with total accumulated demand of Q or less. For every $l \in \Omega$ let us associate a binary variable λ_l equal to 1 if l appears in the solution and 0 otherwise, and a cost c_l for using this route. For every edge $e \in E$ and route $l \in \Omega$ let q_l^e be the number of times that edge e appears in route l. If Ω is restricted to contain only elementary routes (i.e. routes without cycles) then q_l^e is a binary constant, otherwise it can be a general integer. Note that if distances satisfy the triangular inequality, the optimal solution will only contain elementary paths even if Ω is enlarged to contain routes with cycles. In fact, in this case it is always possible to construct, from any solution with cycles, another solution with elementary routes at equal or lower cost. Let us extend the demands to facility nodes by letting $d_v = 0$ for every $v \in I$. A valid formulation for the CLRP is

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{l \in \Omega} c_l \lambda_l \tag{SPF}$$

subject to

$$\sum_{l \in \Omega} \sum_{e \in \delta(\{j\})} q_l^e \lambda_l = 2 \qquad j \in J \tag{2.10}$$

$$\sum_{l \in \Omega_i} \sum_{\{h,j\} \in E} (d_h + d_j) q_l^{\{h,j\}} \lambda_l \leq 2 b_i z_i \qquad i \in I \tag{2.11}$$

$$z_i \in \{0,1\} \qquad i \in I \tag{2.12}$$

$$\lambda_l \geq 0 \text{ and integer} \qquad l \in \Omega. \tag{2.13}$$

2.2.2 Exact algorithms

In this section we describe three algorithms that aim to solve the CLRP exactly. The first is a branch-and-cut method proposed by Belenguer et al. [19] based on the

two-index vehicle-flow formulation augmented by several families of valid inequalities. The second is a branch-and-price method developed by Akca et al. [4] based on the set-partitioning formulation. The third method is a branch-and-cut-and-price algorithm developed by Baldacci et al. [16] based on the set-partitioning formulation augmented by several families of valid inequalities.

2.2.2.1 Branch-and-cut

Belenguer et al. [19] introduced several families of inequalities that are shown to be valid for formulation (VF2). These valid inequalities are embedded into a branch-and-cut solver. They include the so-called y-capacity cuts, facility capacity inequalities, path constraints, degree constraints and co-circuit constraints. These inequalities are complemented with some inequalities from the CVRP, such as multistar inequalities, hypotour inequalities or strengthened comb inequalities [101]. For each of the constraints used in their study they introduced efficient separation algorithms based on some greedy criteria as well as exact methods based on maximum-flow computations. The separation strategy is as follows: at the root node relaxation, all families of cuts are separated aggresively. Deeper in the tree, only some cuts are separated and many of them only once at each node. Regarding the branching strategy, they perform strong branching with priority on the location variables z. If all location variables are integer at a given node, then they perform strong branching on the cutsets defined by the y-capacity cuts added to the problem during the root relaxation. Their computational experience shows that their algorithm is very effective for dealing with small and medium size instances of the CLRP with up to 50 customers. However, some instances with 50 instances cannot be solved with their method and the behaviour on larger instances is not reported.

2.2.2.2 Branch-and-price

The branch-and-price algorthm introduced by Akca et al. [4] is based on formulation (SPF). In this formulation, the set Ω is first restricted to contain a limited number of routes. At each iteration of the algorithm, set Ω is enlarged to contain the columns that

have negative reduced costs with respect to the constraints defining problem (SPF). It can be shown that, in fact, this problem can be formulated as a shortest path problem with resource constraints (SPPRC), which has been introduced by Desrosiers et al. [51] for the VRP with time windows. Routes in the optimal solution of the CLRP will contain no cycles, so it is natural to restrict Ω to contain only elementary routes (i.e. routes without cycles). This is a $\mathscr{NP}$-hard problem for which the most efficient known algorithms are based on dynamic programming [23, 27, 49, 55, 128]. An interesting relaxation of this problem is the 2-*cyc*-SPPRC in which routes are allowed to have cycles of length three or more, but cycles containing two customers are forbidden. This problem is known to be solvable in pseudo-polynomial time but the root relaxation of problem (SPF) is weaker when Ω is enlarged to contain routes with cycles. In the algorithm of Akca et al. [4] both variants of the problem were used. Their computational results show that their algorithms do not scale well with the size of the instances, being able to solve only small instances with up to 40 customers in reasonable computing times.

2.2.2.3 Branch-and-cut-and-price

Baldacci et al. [16] introduced a branch-and-cut-and-price algorithm for the CLRP based on formulation (SPF). Their algorithm works in three stages. In the first stage, they relax the problem and solve the resulting relaxation by column generation using the pricing algorithm of Christofides et al. [32]. They use the bound provided by this relaxation to enumerate all the possible configurations of open facilities that could lead to an improvement of the upper bound. Note that for this to make sense, a valid upper bound of the problem must be available in advance. Moreover, a bad quality upper bound has a strong incidence on the performance of this procedure. Once this problem is solved and all the possible configurations of open facilities are available, the resulting MDVRPs are solved by column-and-cut generation in the second stage. They strengthen formulation (SPF) with the inclusion of a strengthened version of the capacity cuts and also clique inequalities. Their algorithm provides strong lower bounds that usually lie below of 1% of the best known solutions. In the final stage, the remaining columns are enumerated following a very similar method to the used by Baldacci et al. [14, 15].

The resulting integer program is then solved by means of a commercial solver. This algorithm has produced the tightest lower bounds in the literature and is succesful for solving instances with up to 199 customers.

2.2.3 Heuristic algorithms

In this section we describe some of the heuristic algorithms for the CLRP. It includes a GRASP algorithm [121], a memetic algorithm with population management [120], a tabu search algorithm [122], a hybrid metaheuristic combining variable neighborhood search (VNS) with integer linear programming techniques [116], a simulated annealing method [147] and an adaptive large neighborhood search algorithm [78].

2.2.3.1 A GRASP algorithm

GRASP (acronym for *greedy randomized adaptive search procedure*) is a simple metaheuristic paradigm that adds randomization to a given deterministic greedy algorithm in order to diversify the search. Prins et al. [121] described the following GRASP algorithm for the CLRP. They introduce what they call an extended Clarke and Wright savings algorithm (ECWSA), which is based on the well known Clarke and Wright savings algorithm (CWSA) [35] for the CVRP. In the ECWSA, customers are sequentially assigned to their closest facility with sufficient capacity, regardless of the setup costs. Once every customer has been assigned and is served by a single-customer route, a merging operator sequentially takes two routes and merges them. Any two routes are considered for merging, regardless if they belong to the same facility or not. The resulting merged route will be assigned to the closest facility, that may be different from the facilities to which they were originally assigned. The mergings are done either following a *best improvement* or *first improvement* criterion. However, only improving moves are accepted. When no further mergings with positive improvement can be found, facilities without any route assigned to them are closed. In the randomized version of this algorithm, the route merging operator does not follow the *best improvement* or *first improvement* rule. Instead, for a certain parameter $\alpha > 0$, at most the best α moves with

positive improvement are kept in memory at any iteration. When finished inspecting all the merging possibilities, a move is randomly taken and performed among those saved. They call this algorithm randomized ECWSA (RECWSA). RECWSA is repeated several times, thus obtaining several different solutions due to the randomization introduced to the algorithm. To every pair of solutions, a path relinking procedure is performed. Path relinking is another metaheuristic that, given a pair of solutions, guides the search towards the straight segment defining these two solutions with the hope of obtaining a better solution in the middle of the path. For two solutions that are very similar, path relinking usually fails in improving the quality of solutions. For that reason, it is often restricted to pairs of solutions that are far given a certain notion of *distance*.

2.2.3.2 A memetic algorithm with population management

Memetic algorithms (MA) are a special case of genetic algorithms in which the intensification phase is performed by local search procedures. Memetic algorithms with population management (MAPM) are a modified version of MA in which a population of elite solutions is maintained at every iteration, and such that the inclusion of a new solution in the population is subject to some diversification parameter Δ. If Δ represents a threshold distance, then a new solution S will be inserted into population $\mathscr{P}$ if the distance from S to $\mathscr{P}$ is greater than or equal to Δ. Otherwise, the solution S is discarded. MAPM algorithms were first introduced to solve the multidimensional 0-1 knapsack problem by Sörensen and Sevaux [137]. In Prins et al. [120], solutions are coded as chromosomes. These chromosomes can be coded using $|I|+|J|$ bytes and then solutions can be obtained from those chromosomes by using the giant tours split procedure [132] by solving a series of shortest path problems. A crossover operator aims to obtain children as a combination of two chromosomes (called the parents). Local search is then applied to the resulting children. A distance function $d(S,T)$ between chromosomes S,T is defined, and for a population $\mathscr{P}$ with $S \notin \mathscr{P}$ the distance is given by $d(S,\mathscr{P}) = \min_{T\in\mathscr{P}} d(S,T)$.

2.2.3.3 A cooperative lagrangean relaxation-granular tabu search

Tabu search is a popular metaheuristic procedure first introduced by Glover and Laguna [64]. It is a descent-like procedure in which, given a neighborhood, the best movement is performed even if it leads to a deterioration of the current solution. In order to avoid cycling, a tabu list keeps track of the last movements performed, so the opposite movements are declared *tabu* during a certain number of iterations. The concept of *granularity* can be used to restrict a neighborhood to contain only a small subset of moves, so as to accelerate the search [141]. The algorithm presented in Prins et al. [122] consists of two main procedures. The first one is a pure CFLP which is heuristically solved by a Lagrangean method. Given a set of routes defining a solution, one creates a set of super-customers. Each super-customer represents the aggregation of all the customers of a route. The sequence of customers is disconnected from the facility and its two endpoints are reconnected so as to create a subtour. The cost of assigning a super-customer to a given facility is given by the routing cost incurred from assigning such subtour to the facility using as endpoints the two consecutive nodes in the subtour that minimize its routing cost. The second procedure corresponds to a pure MDVRP in which the open facilities are fixed in advance. A tabu search procedure is used to find a good routing solution, and granularity is used to accelerate the neighborhood inspection. The authors realized that these two procedures rapidly converge to local optima. In order to escape from this local optimum, a restart procedure is applied in the location phase in which routes are first split into two or more routes. This procedure creates more super-customers with lower demands, thus providing more flexibility for the solution of the resulting CFLP.

2.2.3.4 A variable neighborhood search coupled with ILP-based heuristics

Variable neighborhood search (VNS) [75, 76] is a recently introduced metaheuristic based on a simple observation. Let $\min_x\{f(x) : x \in \mathscr{X}\}$ be a combinatorial problem. Let $\bar{x} \in \mathscr{X}$ be a feasible solution of that problem and $\mathscr{N} = \{N_k(\bar{x})\}_{k=1}^{k_{max}}$ a family of neighborhoods around $\bar{x}$. Given any neighborhood $N_k(\bar{x})$, consider a randomized algorithm

that picks a random point $\bar{x}' \in N_k(\bar{x})$. Then, local search procedures are applied to $\bar{x}'$ to obtain another point $\bar{x}''$. If $f(\bar{x}'') < f(\bar{x})$ then the solution is updated and the procedure is repeated. The algorithm starts from an initial solution $\bar{y} \in \mathscr{X}$ and from the first neighborhood $N_1(\bar{y})$, and neighborhoods are inspected in increasing order of k. Each time a new solution is found, the procedure is restarted with $k = 1$ and continues until that a certain stopping criteria is met. Pirkwieser and Raidl [116] developed a VNS algorithm for the CLRP coupled with the resolution of several integer-linear programs. In their method, a solution $\mathscr{S}$ of the problem (not necessarily feasible in terms of capacities) is given at any iteration. The set of neighborhoods contains 18 different neighborhoods, some of them nested. The idea is to perform VNS during a certain number of iterations or until reaching local optimality. The local search procedures contain the solution of some integer-linear programs, namely a CFLP and a reallocation problem (RM) [48]. The first is performed more often than the second due to the difficulty to solve the latter problems.

2.2.3.5 A simulated annealing algorithm

Simulated annealing (SA) is a popular metaheuristic which includes a simple mechanism to avoid getting trapped in local optima. This procedure owes its name to the annealing process in metallurgy, in which a given piece of metal is heated above its recrystallization temperature, modified in its shape and then cooled. In combinatorial optimization, the SA method was first inroduced by Metropolis et al. [104] and then made popular by Kirkpatrick et al. [83]. Given a combinatorial problem $\min_x\{f(x) : x \in \mathscr{X}\}$, a point $\bar{x} \in \mathscr{X}$ (the incumbent) and a neighborhood $N(\bar{x})$, another point $\bar{x}' \in N(\bar{x})$ is obtained randomly in $N(\bar{x})$. If $f(\bar{x}') < f(\bar{x})$ then the new solution is accepted and the incumbent is updated. Otherwise, the solution will be accepted with a probability given by the Boltzmann function $\exp(-\Delta/kT)$, where $\Delta = f(\bar{x}') - f(\bar{x})$, k is a constant and T is the cooling temperature. Normally, the cooling temperature T decreases in time, so as to lower the probability of accepting worse solutions than the incumbent. At the beginning of the algorithm, T is set to a high value and the search often visits bad solutions with the hope of escaping from local optima, whereas at the end of the algorithm only solu-

tions with costs close to the incumbent will be accepted. Yu et al. [147] presented a SA algorithm for the CLRP. In their method, several neighborhoods are considered, namely *insertion*, *swap* and *2-opt* moves. At each iteration, one of these moves is chosen with an uniform probability of $1/3$.

2.2.3.6 An adaptive large neighborhood search algorithm

Large neighborhood search (LNS) is a recently introduced heuristic [134]. It is based on the concept of *destroy and repair*, in which two sets of operators $\mathscr{D} = \{D_k\}_{k=1}^{k_D}$ (destroying operators) and $\mathscr{R} = \{R_k\}_{k=1}^{k_R}$ (repairing operators) are considered. At each iteration, a pair of operators $(D,R) \in \mathscr{D} \times \mathscr{R}$ is randomly selected and applied sequentially. The *destroy* operator D takes a complete solution and returns an incomplete solution, in which certain edges or nodes are missing or disconnected from the rest of the solution. The *repair* operator R is then applied to the incomplete solution and returns a new complete solution. This procedure is repeated until a certain criterion is met. In the adaptive LNS (ALNS) the *destroy* and *repair* operators are ranked at every iteration according to their success to improve solutions during previous iterations. In subsequent iterations, the pair of operators (D,R) chosen to *destroy* and *repair* a given solution are taken not using an uniform distribution but rather a probability function that gives a higher probability to operators with higher rankings. This method has been successfully applied to different classes of vehicle routing problems [117, 129]. Hemmelmayr et al. [78] developed an ALNS heuristic for the two-echelon VRP (2E-VRP), of which the CLRP arises as a particular case. Destroy and repair operators contain moves involving insertion, deletion or swapping of facilities, customers or full routes. They complement the search with the use of some efficient local search procedures that are applied at some selected parts of their algorithm.

CHAPTER 3

BRANCH-AND-CUT ALGORITHMS

Notes about the chapter

The contents of this chapter correspond to those of the article entitled *A Computational Comparison of Flow Formulations for the Capacitated Location-Routing Problem*, co-authored with Professors Jean-François Cordeau and Bernard Gendron, which has been submitted for publication to Discrete Optimization (ISSN: 1572-5286), in July 2011. Preliminary results have also been presented in several conferences and workshops, including the *VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, in Buenos Aires, Argentina (2009) and *Optimization Days 2009*, in Montréal, Canada (2009).

A Computational Comparison of Flow Formulations for the Capacitated Location-Routing Problem

Claudio Contardo[1,3,*], Jean-François Cordeau[2,3], Bernard Gendron[1,3]

[1]Département d'informatique et de recherche opérationnelle, Université de Montréal
C.P. 6128, succ. Centre-ville, Montréal (PQ) Canada H3C 3J7

[2]Canada Research Chair in Logistics and Transportation and HEC Montréal
3000 chemin de la Côte-Sainte-Catherine, Montréal (PQ) Canada H3T 2A7

[3]Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT)
C.P. 6128, succ. Centre-ville, Montréal (PQ) Canada H3C 3J7

In this paper we present a computational comparison of four different flow formulations for the capacitated location-routing problem. We introduce three new flow formulations for the problem, namely a two-index two-commodity flow formulation, a three-index vehicle-flow formulation and a three-index two-commodity flow formulation. We also consider the known two-index vehicle-flow formulation of Belenguer et al. [19] and extend it by considering new families of valid inequalities and separation algorithms. We introduce new branch-and-cut algorithms for each of the formulations and compare them on a wide number of instances. Our results show that compact formulations can produce tight gaps and solve many instances quickly, whereas three-index formulations scale better in time.

Key words: location-routing, flow formulations, branch-and-cut.
AMS Classification Codes: 90C27, 90C57.
[*]*Corresponding author.* Tel: +1-514-343-7575. Fax: +1-514-343-7121.

3.1 Introduction

In the *Capacitated Location-Routing Problem* (CLRP) we are given a set I of potential facility locations and a set J of customers. The problem consists in selecting a subset

of facilities and in designing vehicle routes around these facilities so that every customer is visited exactly once. Each facility $i \in I$ has a capacity b_i and a fixed cost f_i. The fleet is unlimited and each vehicle has a capacity Q. Each customer $j \in J$ has a demand d_j. We define a graph $G = (V,E)$ where $V = I \cup J$ is the vertex set and E is the edge set. With every edge $\{i, j\} \in E$ is associated a cost c_{ij} for using edge $\{i, j\}$. Each route must start from and return to the same selected facility, and the sum of the demands of the customers served along a route cannot exceed Q. In addition, the total demand of the customers served in routes from facility i cannot exceed b_i. The objective consists in minimizing the sum of the routing costs and the fixed costs associated with the selected facilities.

A three-index mixed-integer programming formulation for the CLRP was introduced by Perl and Daskin [115] for the general case of an asymmetric network, heterogeneous vehicles and heterogeneous facilities. Its linear programming relaxation does not, however, provide lower bounds that are tight enough to be used within a branch-and-cut algorithm. Laporte et al. [91] proposed the first two-index vehicle-flow formulation for the LRP with uncapacitated facilities (ULRP). They have considered vehicle capacity cuts (CC) as well as chain barring constraints (CBC) and, by means of a branch-and-bound algorithm, were able to solve small size instances. Based on this work, Belenguer et al. [19] recently proposed a two-index integer programming formulation for the CLRP, providing strengthened versions of the CC and the CBC. They also introduced a new version of the facility capacity inequalities (FCI) and other constraints such as co-circuit constraints and depot degree constraints. The lower bounds obtained by their algorithm are very tight and suggest that by improving the separation algortithms as well as developing new families of valid inequalities, the cutting-plane approach would lead to a successful methodology for solving medium or even large size instances of the CLRP. Akca et al. [4] have introduced a mixed set partitioning / knapsack formulation by doing a Dantzig-Wolfe decomposition of the three-index formulation that is solved by means of a branch-and-price method. The pricing problem consists in finding elementary paths of minimum reduced cost under capacity constraints. The lower bounds obtained by their algorithm show a significant improvement with respect to those obtained by the algo-

rithms based on the two-index vehicle-flow formulation. More recently, Baldacci et al. [16] have proposed a branch-and-cut-and-price algorithm. They apply two bounding procedures to compute a tight lower bound, followed by the optimal solution of a small number of multiple depot vehicle routing problems (MDVRP). They provide a strengthened version of the CC as well as clique inequalities for the set-partitioning problem. Their algorithm improves the lower bounds of the previous approaches and solves to optimality instances with up to 199 customers and 14 facilities.

The CLRP is known to be $\mathscr{NP}$-hard, as it combines (and includes as particular cases) both the Capacitated Vehicle Routing Problem (CVRP) and the Capacitated Facility Location Problem (CFLP). Authors have thus focused their attention on the development of heuristic methods to find good quality solutions in reasonable computing times. Most of these heuristics are based on decomposition techniques that solve a location (design) and a routing (operational) sub-problem. Depending on whether the algorithm iterates between the two subproblems, we distinguish between sequential algorithms [115] and iterative algorithms [77, 97, 99, 146]. Tuzun and Burke [142] decompose the problem into a location and a routing subproblem, but the location decisions at each iteration only consider the opening of new facilities or the swapping of two already open facilities, so the whole algorithm rapidly converges to a local optimum. Other heuristics include memetic algorithms [120] or Lagrangian heuristics [113].

The main contributions of this paper can be summarized as follows:

i. We introduce three new formulations based on vehicle flows and commodity flows, which are proven to dominate, in terms of the linear relaxation lower bound, the two-index vehicle-flow formulation of Belenguer et al. [19] at the expense of adding more variables.

ii. We derive two new families of multistar inequalities from the commodity-flow formulations and introduce separation algorithms for using them inside the vehicle-flow formulations.

iii. We introduce several new families of valid inequalities for the formulations introduced in this paper, and strengthen several of the existing ones.

iv. We introduce new, efficient separation algorithms for the inequalities used in our algorithms, which in many cases generalize those introduced by Belenguer et al. [19].

v. We perform a computational study comparing each of the formulations on a large number of instances and discuss their advantages and disadvantages.

The rest of the paper is organized as follows. In Section 3.2 we first describe the two-index formulation introduced by Belenguer et al. [19]. We then introduce the three new formulations based on vehicle flows and commodity flows. We prove that for the case of the comodity-flow formulations, some new classes of multistar inequalities are implied. In Section 3.3 we present both existing and new families of valid inequalities for the CLRP. In Section 3.4 we begin by introducing a general heuristic for generating cuts, and we then introduce the separation algorithms for each of the valid inequalities introduced in the paper. In Section 3.5 we describe the branch-and-cut algorithms used in our experiments by specifying the separation and branching strategies. In Section 3.6 we present a computational study performed after running our algorithms on several families of instances. This is followed by the conclusions in Section 3.7. To improve clarity, we provide in a separate section the proofs of the lemmas and propositions introduced in Sections 3.2, 3.3 and 3.4.

3.2 Mathematical Formulations

In this section we first describe the two-index formulation introduced by Belenguer et al. [19] for the CLRP with a homogeneous fleet and symmetric costs. We then introduce three new formulations based on vehicle flows and two-commodity flows.

3.2.1 A two-index vehicle-flow formulation [19]

We first introduce the notation that we will use throughout the article and then present the formulation itself.

Let $G = (V,E)$ be an undirected graph, where $V = I \cup J$ and $E = \{\{v_i,v_j\} : v_i,v_j \in V\} \setminus I \times I$. For every subset $U \subseteq V$, we define $E(U) = \{\{u,w\} \in E : u,w \in U\}$, and

$\delta(U) = \{\{u,w\} \in E : u \in U, w \notin U\}$. For every pair of disjoint subsets U and W, let also $(U:W) = \{\{u,w\} \in E : u \in U, w \in W\}$. With every edge $e \in \delta(I)$ are associated two binary variables: x_e equal to 1 iff edge e is used once, and y_e equal to 1 iff edge e is used twice. With every edge $e \in E(J)$ is associated a binary variable x_e equal to 1 iff edge e is used. For every facility $i \in I$, let z_i be a binary variable equal to 1 iff facility i is selected. For a given edge set $F \subseteq E$ we define $x(F) = \sum_{e \in F} x_e$ and $y(F) = \sum_{e \in F} y_e$ (if $F \subseteq \delta(I)$). For a given subset $S \subseteq J$ of customers, we define $d(S) = \sum_{j \in S} d_j$, and a constant $r(S) = \lceil d(S)/Q \rceil$ which is a lower bound on the number of vehicles required to satisfy the demand of customers in S. Finally, we define $\overline{S} = J \setminus S$. The CLRP can then be formulated as the following integer program.

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{e \in E} c_e x_e + 2 \sum_{e \in \delta(I)} c_e y_e \tag{VF2}$$

subject to

$$x(\delta(j)) + 2y(I:\{j\}) = 2 \qquad j \in J \qquad (3.1)$$

$$x(\delta(S)) + 2y(I:S) \geq 2r(S) \qquad S \subseteq J, |S| \geq 2 \qquad (3.2)$$

$$x_{ij} + y_{ij} \leq z_i \qquad i \in I, j \in J \qquad (3.3)$$

$$x(I:\{j\}) + y(I:\{j\}) \leq 1 \qquad j \in J \qquad (3.4)$$

$$x((I \setminus \{i\}) \cup \overline{S} : S) + 2y(I \setminus \{i\} : S) \geq 2 \qquad i \in I, S \subseteq J, d(S) \geq b_i \qquad (3.5)$$

$$x(\delta(S)) \geq 2(x(\{h\} : I') + x(\{j\} : I \setminus I')) \qquad S \subseteq J, |S| \geq 2, h, j \in S, I' \subset I \qquad (3.6)$$

$$z_i \in \{0,1\} \qquad i \in I \qquad (3.7)$$

$$x_e \in \{0,1\} \qquad e \in E \qquad (3.8)$$

$$y_e \in \{0,1\} \qquad e \in \delta(I). \qquad (3.9)$$

Demand constraints (3.1) impose that every customer vertex be visited once and also act as flow conservation equations. Constraints (3.2) are the capacity cuts (CC) which play a dual role: they forbid tours disconnected from facilities as well as tours

serving a demand larger than Q. Constraints (3.3) ensure that there is no outgoing flow from unselected facilities. Constraints (3.4) forbid single-customer routes to be linked to two different facilities. Constraints (3.5) are the facility capacity inequalities (FCI). They forbid the existence of a set of routes leaving from a given facility i and serving a demand higher than b_i. Constraints (3.6) are the path constraints (PC) that prevent the route of a vehicle from joining two different facilities. These constraints are not valid when $|S| = 1$ and they are thus complementary to constraints (3.4).

Unlike in traditional CVRP formulations, two sets of variables (x and y) are associated with the edges in $\delta(I)$. One can in fact check that if these variables are replaced with the aggregated variables $\bar{x}_e = x_e + 2y_e$, single-customer routes linked to two different facilities can no longer be correctly eliminated as we do with constraints (3.4).

3.2.2 A three-index vehicle-flow formulation

Due to their large number of variables, three-index formulations for vehicle routing problems have limited practical interest. In these formulations, two indices represent a certain edge while the third index indicates which vehicle uses this edge. These formulations naturally provide tighter bounds than their two-index counterparts when augmented by all of the known valid inequalities. However, they also present a lot of symmetry that makes them of little use within a branch-and-bound framework. We introduce a three-index formulation which does not suffer from this issue. Indeed, we use the third index to specify the facility from which the edge is being visited. Symmetry is then not an issue because switching two facilities does not provide an alternate equivalent solution, either because of feasibility (facility capacities may not be the same) or costs (switching routes from one facility to another usually produces a change in either the routing costs or the fixed costs). Using the same notation as for the two-index vehicle-flow formulation, we define binary variables x_e^i equal to 1 iff edge e is used once by a vehicle being from facility $i \in I$ (naturally $x_{lj}^i = 0$ if $l, i \in I, l \neq i$). We also let y_{ij} be a binary variable equal to 1 iff edge $e = \{i, j\}$ is used twice (for single-customer routes) by a vehicle linked to facility i. We let u_{ij} be a binary variable equal to 1 iff customer j is served from facility i. Let us define the following notation. For an edge subset $F \subseteq E$ and a facility subset

$H \subseteq I$ we let $x^H(F) = \sum_{i \in H} \sum_{e \in F} x^i_e$, and if $H = \{i\}$ is a singleton we let $x^i(F) = x^{\{i\}}(F)$. The formulation is the following,

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{e \in E} c_e x^i_e + 2 \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \tag{VF3}$$

subject to

$$x^i(\delta(\{j\})) + 2y_{ij} = 2u_{ij} \qquad i \in I, j \in J \tag{3.10}$$

$$x^i(\delta(S)) + 2y(\{i\} : S) \geq \frac{2}{Q} \sum_{j \in S} d_j u_{ij} \qquad i \in I, S \subseteq J, |S| \geq 2 \tag{3.11}$$

$$\sum_{j \in J} d_j u_{ij} \leq b_i z_i \qquad i \in I \tag{3.12}$$

$$\sum_{i \in I} u_{ij} = 1 \qquad j \in J \tag{3.13}$$

$$x^i_{ij} + y_{ij} \leq u_{ij} \leq z_i \qquad i \in I, j \in J \tag{3.14}$$

$$z_i \in \{0,1\} \qquad i \in I \tag{3.15}$$

$$x^i_e \in \{0,1\} \qquad i \in I, e \in E \tag{3.16}$$

$$y_e \in \{0,1\} \qquad e \in \delta(I) \tag{3.17}$$

$$u_{ij} \in \{0,1\} \qquad i \in I, j \in J. \tag{3.18}$$

Constraints (3.10) are a disaggregated form of the degree equations (3.1), whereas constraints (3.11) are a disaggregated form of the capacity inequalities (3.2). Constraints (3.12) are the facility capacity inequalities. Constraints (3.13) are the assignment constraints of customers to facilities. Constraints (3.14) link the assignment variables with the flow and location variables.

3.2.3 A two-index two-commodity flow formulation

Each facility node $i \in I$ is considered as a source of flow, to which we consider an additional sink node i'. Let us denote the set of sink facility nodes as I', and consider the

augmented *undirected* graph $\overline{G} = (\overline{V}, \overline{E})$ with $\overline{V} = V \cup I'$ and $\overline{E} = E \cup \{e = \{i', j\} : i' \in I', j \in J\}$. A route starting and ending at a facility i in the original graph will be mapped to a flow in the new graph starting at i and arriving to i'. For this purpose, let us introduce the following set of continuous variables. For every edge $e = \{i, j\} \in \overline{E}$, we define an arc variable w_{ij} which denotes the amount of flow traversing edge *e if e is traversed from node i to node j*, and w_{ji} represents the remaining capacity on the vehicle traversing this edge. If the trip is done in the opposite direction the roles of w_{ij} and w_{ji} are reversed. To take into account the orientation defined by these new variables, we define for every set $U \subseteq V$, $w(\delta^+(U)) = \sum_{u \in U, v \notin U} w_{uv}$, $w(\delta^-(U)) = \sum_{u \in U, v \notin U} w_{vu}$. We keep variables y for single-customer trips, while variables w are only used for multiple-customer routes (i.e., routes serving two or more customers). The following set of constraints are thus valid for the CLRP

$$w(\delta^-(\{j\})) - w(\delta^+(\{j\})) + 2d_j y(I : \{j\}) = 2d_j \qquad j \in J \qquad (3.19)$$

$$w(\delta^+(\{i\})) + \sum_{j \in J} d_j y_{ij} \leq b_i z_i \qquad i \in I \qquad (3.20)$$

$$w(\delta^+(\{i'\})) = Qx(\delta(\{i\})) \qquad i' \in I' \qquad (3.21)$$

$$w_{ij} + w_{ji} = Qx_{ij} \qquad \{i, j\} \in \overline{E} \qquad (3.22)$$

$$w_{ij}, w_{ji} \geq 0 \qquad \{i, j\} \in \overline{E}. \qquad (3.23)$$

Now, vehicle capacities and facility capacities are implied by (3.19)-(3.23). A valid formulation for the CLRP is given by

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{e \in E} c_e x_e + 2 \sum_{e \in \delta(I)} c_e y_e \qquad \text{(CF2)}$$

subject to (3.1), (3.3)-(3.4), (3.6)-(3.9), (3.19)-(3.23).

Baldacci et al. [13] proved that the following flow inequalities (FI) are valid for the two-index two-commodity flow formulation of the CVRP:

$$(Q-d_j)w_{ij}-d_jw_{ji}\geq 0 \qquad \{i,j\}\in\overline{E} \tag{3.24}$$

$$(Q-d_i)w_{ji}-d_iw_{ij}\geq 0 \qquad \{i,j\}\in\overline{E}. \tag{3.25}$$

It is straightforward to check that they also are for the CLRP. As stated by the following proposition, they also imply the following y-generalized large multistar inequalities (y-GLM),

Proposition 3.2.1. *The following y-generalized large multistar inequalities (y-GLM) are implied by formulation* (CF2) *when augmented with the flow inequalities (FI):*

$$x(\delta(S))+2\sum_{j\in S}\frac{d_j}{Q}y(I:\{j\})\geq\frac{2}{Q}\left(d(S)+\sum_{\substack{h\in S\\ j\notin S}}d_jx_{hj}\right). \tag{3.26}$$

The generalized large multistar inequalities (GLM) which are valid for the CLRP differ from these inequalities in the coefficients d_j/Q multiplying the terms $y(I:\{j\})$ which are replaced by 1. Therefore, the y-GLM dominate the GLM.

3.2.4 A three-index two-commodity flow formulation

Let us consider the three-index vehicle-flow formulation (VF3). As for the previous formulation, we consider the augmented graph $\overline{G}$ and we use variables w^i_{hj}, w^i_{jh} for the flow traversing edge $\{h,j\}$ from facility i and for the remaining capacity in the vehicle, respectively. We keep variables y_{ij} for single-customer routes. For a facility $i\in I\cup I'$ and a node subset $U\subseteq\overline{V}$ we denote $w^i(\delta^+(U))=\sum_{u\in U,v\notin U}w^i_{uv}$, $w^i(\delta^-(U))=\sum_{u\in U,v\notin U}w_{vu}$. Formulation (VF3) can thus be augmented by adding these variables and the following set of constraints:

$$w^i(\delta^-(\{j\})) - w^i(\delta^+(\{j\})) + 2d_j y_{ij} = 2d_j u_{ij} \qquad i \in I, j \in J \qquad (3.27)$$

$$w^i(\delta^+(\{i\})) + \sum_{j \in J} d_j y_{ij} \leq b_i z_i \qquad i \in I \qquad (3.28)$$

$$w^i(\delta^+(\{i'\})) = Q x^i(\delta(\{i\})) \qquad i' \in I' \qquad (3.29)$$

$$w^i_{hj} + w^i_{jh} = Q x^i_{hj} \qquad i \in I, \{h,j\} \in \overline{E} \qquad (3.30)$$

$$w^i_{hj}, w^i_{jh} \geq 0 \qquad i \in I, \{h,j\} \in \overline{E}. \qquad (3.31)$$

The new formulation for the CLRP is the following

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{e \in E} c_e x^i_e + 2 \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \qquad \text{(CF3)}$$

subject to (3.10), (3.13)-(3.18), (3.27)-(3.31).

Note that the following disaggregated flow inequalities (DFI) are valid for this formulation

$$(Q - d_j) w^i_{hj} - d_j w^i_{jh} \geq 0 \qquad i \in I, \{h,j\} \in \overline{E} \qquad (3.32)$$

$$(Q - d_h) w^i_{jh} - d_h w^i_{hj} \geq 0 \qquad i \in I, \{h,j\} \in \overline{E}. \qquad (3.33)$$

As a consequence of the extra variables added with respect to the two-index two-commodity flow formulation, this one also implies the following y-location routing generalized large multistar inequalities (y-LRGLM),

Proposition 3.2.2. *The following y-location routing generalized large multistar inequalities (y-LRGLM) are implied by formulation* (CF3) *plus the disaggregated flow inequalities (DFI).*

$$x^{I \setminus H}(\delta(S)) + 2 \sum_{j \in S} \frac{d_j}{Q} y(I \setminus H : \{j\}) \geq \frac{2}{Q} \left(\sum_{i \in I \setminus H} \sum_{j \in S} d_j u_{ij} + \sum_{\substack{h \in S \\ j \notin S}} d_j x^{I \setminus H}_{hj} \right). \qquad (3.34)$$

Remark Note that the particular case $H = \emptyset$ corresponds to the y-GLM (3.26).

3.3 Valid Inequalities

In this section we consider several families of valid inequalities that can be used to strengthen the linear relaxation of the previous formulations. We first describe known inequalities and then introduce new families of valid inequalities.

3.3.1 Known valid inequalities

In this subsection we describe valid inequalities that are already known for the CLRP. These include constraints for the CVRP such as framed capacity inequalities (FrCI), strengthened comb inequalities (SCI), multistar inequalities (MSI), hypotour inequalities (HYP), y-capacity cuts (y-CC), strengthened facility capacity inequalities (SFCI), co-circuit constraints (CoCC) and facility degree constraints (FDC). For details on each of these inequalities we refer to Lysgaard et al. [101] and to Belenguer et al. [19].

3.3.1.1 Inequalities for the CVRP

If nodes in I are contracted into a single node, the resulting problem can be seen as a CVRP instance. If a cut valid for the CVRP is such that the coefficients of the edges joining the depot to customers do not vary with the depot (as the distance, for instance), this cut remains valid for the CLRP by considering this contracted graph. This is the case for all of the known valid inequalities, in particular, strengthened comb inequalities, multistar inequalities, generalized large multistar inequalities, framed capacity inequalities and hypotour inequalities [101]. We add them all except for the generalized large multistar inequalities which are replaced by the y-GLM (3.26).

3.3.1.2 y-Capacity cuts [19]

Let us consider constraints (3.2) for a given customer set S. Additionally, suppose that we are given a customer subset S' satisfying $r(S \setminus S') = r(S)$. The following constraint, called y-capacity cut or simply y-CC, is valid for the CLRP and dominates (3.2):

$$x(\delta(S)) + 2y(I : S \setminus S') \geq 2r(S). \tag{3.35}$$

For the proof that these constraints are valid, we refer to Belenguer et al. [19].

3.3.1.3 Strengthened facility capacity inequalities [19]

For a given facility set I', let us denote $b(I') = \sum_{i \in I'} b_i$. Belenguer et al. [19] proposed the following two strengthenings for inequalities (3.5). Let $S \subseteq J$ and $i \in I$ be as in inequalities (3.5). Let $I' \subset I$ be a subset of facilities such that $i \in I'$. If a subset $S' \subset S$ is such that $d(S \setminus S') > b(I')$ then the following strengthened facility capacity inequality (SFCI) is valid for the CLRP:

$$x((I \setminus I') \cup \bar{S} : S) + 2y(I \setminus I' : S \setminus S') \geq 2. \tag{3.36}$$

Let $r(S,I') = \lceil (d(S) - b_{I'})/Q \rceil$ be a lower bound on the number of vehicles needed to serve the demand of customers in S from facilities other that those in I'. Note that although $r(\cdot)$ and $r(\cdot,\cdot)$ represent different quantities, the overloaded notation satisfies $r(S,\emptyset) = r(S)$ for every $S \subseteq J$. The following inequality is valid for the CLRP:

$$x((I \setminus I') \cup \bar{S} : S) + 2y(I \setminus I' : S) \geq 2r(S,I' \setminus \{i\}) + 2z_i(r(S,I') - r(S,I' \setminus \{i\})). \tag{3.37}$$

We call these inequalities the effective SFCI (ESFCI). For the validity of these inequalities we refer again to Belenguer et al. [19].

3.3.1.4 Co-circuit constraints [19]

The co-circuit constraints (CoCC) state that the graph resulting from the deletion of the y variables must still have an even number of edges. They can be written as

$$x(\delta(S) \setminus F) \geq x(F) - |F| + 1 \tag{3.38}$$

for $S \subseteq J$, $F \subseteq \delta(S)$ and $|F|$ odd.

3.3.1.5 Facility degree constraints [19]

The facility degree constraints (FDC) are valid under the assumption that the triangle inequality holds for the edge distances. It states the sub-optimality of solutions in which two or more vehicles serve a given set of customers if these customers can be served by fewer vehicles (thus saving travel time). For single-customer routes they can be written as

$$y(i:S) \leq z_i \tag{3.39}$$

$\forall S \subseteq J$ such that $d_h + d_j \leq Q, \forall h \neq j \in S, \forall i \in I$. For general routes, they can be written as

$$2y(i:S) + x(i:S) + x(E(S)) \leq 2z_i + |S| - 1 \tag{3.40}$$

$\forall i \in I, \forall S \subseteq J, r(S) = 1$.

3.3.2 New valid inequalities

In this subsection we introduce new families of valid inequalities for the CLRP. These include strengthened versions of the SFCI, ESFCI, location-routing comb inequalities (LRCOMB), location-routing generalized large multistar inequalities (LRGLM) and flow-assignment inequalities (FAI), all of which are valid for the two-index formulations and by extension for the three-index formulations as well. Moreover, we strengthen some of these inequalities for the case of the three-index formulations, and add some novel classes of inequalities that cannot be derived from the former.

3.3.2.1 Flow-assignment inequalities

It is easy to check that the following inequalities are valid for the two-index and three index commodity-flow formulations, respectively:

$$w_{ij} + w_{ji} \leq Q \qquad \{i,j\} \in \overline{E} \tag{3.41}$$

$$w^l_{ij} + w^l_{ji} \leq Q \qquad l \in I, \{i,j\} \in \overline{E}. \tag{3.42}$$

However, they can be strengthened as a consequence of the following two observations. First, for every edge $e = \{i, j\} \in E$, at least one node i or j belongs to J. For that node, say j, it cannot happen at the same time that edge $\{i, j\}$ is used by a vehicle serving two or more customers and j is served by a single-cutomer route. Thus, the following flow-assignment inequalities (FAI) are valid for the CLRP:

$$x_{ij} + y(I : \{j\}) \leq 1 \qquad j \in J, \{i, j\} \in E. \tag{3.43}$$

$$w_{ij} + w_{ji} + Qy(I : \{j\}) \leq Q \qquad j \in J, \{i, j\} \in \overline{E}. \tag{3.44}$$

$$x^l_{ij} + y_{lj} \leq u_{lj} \qquad l \in I, j \in J, \{i, j\} \in E. \tag{3.45}$$

$$w^l_{ij} + w^l_{ji} + Qy_{lj} \leq Qu_{lj} \qquad l \in I, j \in J, \{i, j\} \in \overline{E}. \tag{3.46}$$

In the case of the three-index formulations, constraints (3.45)-(3.46) impose a strong relationship between the flow variables and the customers assignments. Indeed, if a customer is not assigned to a given facility, then all flow variables associated to the corresponding facility and linked to that customer are automatically set to 0.

3.3.2.2 Disaggregated co-circuit constraints

The co-circuit constraints (3.38) ensure that an even number of edges will traverse a given customer subset $S \subseteq J$. This is in particular valid when restricted to the edges used by some facility. Thus, for the particular case of the three-index formulations the following disaggregated co-circuit constraints (DCoCC) are valid for the CLRP:

$$x^i(\delta(S) \setminus F) \geq x^i(F) - |F| + 1 \qquad i \in I, S \subseteq J, F \subset \delta(S), |F| \text{ odd}. \tag{3.47}$$

Proposition 3.3.1. *Constraints* (3.47) *are valid for the CLRP.*

3.3.2.3 Disaggregated facility degree constraints

Using the same reasoning as for the CoCC, the facility degree constraints (3.40) also have their disaggregated counterpart. Indeed, if distances satisfy the triangle inequality,

the following inequalities are valid for the three-index formulations of the CLRP:

$$x^i(i:S)+2y(i:S)+x^i(E(S))\leq\sum_{j\in S}u_{ij}+z_i \qquad i\in I,S\subseteq J,d(S)\leq Q. \tag{3.48}$$

Proposition 3.3.2. *Constraints* (3.48) *are valid for the CLRP.*

3.3.2.4 Strengthened facility capacity inequalities

Let us consider inequalities (3.36) for given $S\subseteq J$ and $I'\subseteq I$. If $S'\subset S$ is such that $r(S\setminus S',I')=r(S,I')$ let us consider the following inequality:

$$x((I\setminus I')\cup\overline{S}:S)+2y(I\setminus I':S\setminus S')\geq 2r(S,I'). \tag{3.49}$$

Proposition 3.3.3. *Constraints* (3.49) *are valid for the CLRP.*

As these constraints dominate (3.36), we will now refer to these inequalities as SFCI. These constraints are valid for all the formulations studied in this paper. However, for the three-index case they can be strengthened to the following constraints:

$$x^{I\setminus I'}(\delta(S))+2y(I\setminus I':S\setminus S')\geq 2r(S,I'). \tag{3.50}$$

3.3.2.5 Effective strengthened facility capacity inequalities

Let us consider constraints (3.37) for given S,I' and $i\in I'$. Suppose that $S'\subseteq S$ is such that $r(S\setminus S',I')=r(S,I')$ and $r(S\setminus S',I'\setminus\{i\})=r(S,I'\setminus\{i\})$. Then, the following inequality is valid for the CLRP and dominates (3.37):

$$x((I\setminus I')\cup\overline{S}:S)+2y(I\setminus I':S\setminus S')\geq 2r(S,I'\setminus\{i\})+2z_i(r(S,I')-r(S,I'\setminus\{i\})). \tag{3.51}$$

As this inequality dominates (3.37), we will refer to it as the ESFCI.

Proposition 3.3.4. *Constraints* (3.51) *are valid for the CLRP.*

Just as with the SFCI, for the three-index case these inequalities can be strengthened to the following set of inequalities

$$x^{I\setminus I'}(\delta(S)) + 2y(I \setminus I' : S \setminus S') \geq 2r(S, I' \setminus \{i\}) + 2z_i(r(S, I') - r(S, I' \setminus \{i\})). \quad (3.52)$$

Remark Constraints SFCI and ESFCI do not dominate each other. However, in practice, we have verified that for fractional values of the z variables the ESFCI have a more significant impact on the lower bound than the SFCI. Conversely, when location variables are fixed to either 0 or 1, constraints SFCI start playing an important role. Because of that, at every node of the branching tree we separate constraints ESFCI for facilities i such that $0 < z_i \leq 0.85$ and constraints SFCI for every i such that $0.75 < z_i \leq 1$. The role of every cut is complementary: constraints ESFCI help to stregthen the lower bound and hopefully to prune nodes close to the root, while constraints SFCI start dominating the ESFCI deeper in the tree.

3.3.2.6 Location-routing comb inequalities

Comb inequalities were developed by Chvátal [33] for the symmetric traveling salesman problem (STSP) and have since then received considerable attention in the literature [68, 90, 101]. In particular, stronger versions have been proposed for the CVRP that take advantage of the vehicle capacities. In what follows we develop a new family of inequalities that are shown to be valid for the CLRP and include some of the earlier inequalities as special cases. Let sets $H \subseteq V$ (the *handle*), $\Pi = (T_j^1)_{j=1}^{s_1} \cup (T_j^2)_{j=1}^{s_2} \subseteq \mathscr{P}(V)$ (the *teeth*) be such that

i. $|H \cap T| \geq 1 \quad T \in \Pi$

ii. $|T \setminus H| \geq 1 \quad T \in \Pi$

iii. $|T \cap U| = 0 \quad T, U \in \Pi$

iv. $|H \cap I| = 0$

v. $|T_j^1 \cap I| \geq 1 \quad 1 \leq j \leq s_1$

vi. $|T_j^2 \cap I| = 0 \quad 1 \leq j \leq s_2$

For notational simplicity, for every k, j we denote $S_j^k = T_j^k \cap J$. If $k = 1$, we also denote $I_j = T_j^1 \cap I$. Let $s_1' < s_1$ and suppose that for each $j \in \{1, \ldots, s_1'\}$ we also distinguish a special facility $i_j \in I_j$ that we call *effective*. For every set $U \subseteq V = I \cup J$ let us denote

$$\overline{x}(E(U)) = \begin{cases} x(E(U)) & \text{if } U \cap I = \emptyset \\ x(E(U \setminus I)) + x(U \cap I : U \setminus I) + 2y(U \cap I : U \setminus I) & \text{if } U \cap I \neq \emptyset. \end{cases}$$

Let $\alpha\overline{x} = \overline{x}(E(H)) + \sum_{k=1}^{2} \sum_{j=1}^{s_k} \overline{x}(E(T_j^k))$. Define the following constants:

$$\widehat{r}(H, T_j^k) = \begin{cases} r(S_j^1, I_j \setminus \{i_j\}) + r(S_j^1 \setminus H, I_j \setminus \{i_j\}) + r(S_j^1 \cap H) & \text{if } k = 1, 1 \leq j \leq s_1' \\ r(S_j^1, I_j) + r(S_j^1 \setminus H, I_j) + r(S_j^1 \cap H) & \text{if } k = 1, s_1' < j \leq s_1 \\ r(S_j^2) + r(S_j^2 \setminus H) + r(S_j^2 \cap H) & \text{if } k = 2, 1 \leq j \leq s_2 \end{cases}$$

$$\Lambda(H, T_j^1) = r(S_j^1, I_j \setminus \{i_j\}) + r(S_j^1 \setminus H, I_j \setminus \{i_j\}) - r(S_j^1, I_j) - r(S_j^1 \setminus H, I_j) \quad 1 \leq j \leq s_1$$

$$\widehat{r}(H, \Pi) = \sum_{k=1,2} \sum_{1 \leq j \leq s_k} \widehat{r}(H, T_j^k).$$

If $\Lambda(H, T_j^1)$ is *even* for every $1 \leq j \leq s_1'$ and $\widehat{r}(H, \Pi)$ is *odd*, the associated location-routing comb inequality (LRCOMB) is

$$\alpha\overline{x} - \tfrac{1}{2}\Big[\sum_{1 \leq j \leq s_1} \big(x(I_j : J) + 2y(I_j : J)\big) + \sum_{1 \leq j \leq s_1'} z_{i_j} \Lambda(H, T_j^1) \Big] \leq |H| + \sum_{k=1}^{2} \sum_{j=1}^{t} |S_j^k| - \left\lceil \tfrac{1}{2} \widehat{r}(H, \Pi) \right\rceil. \quad (3.53)$$

Proposition 3.3.5. *The location-routing comb inequality* (3.53) *is valid for the CLRP.*

Remark 1. For the sake of clarity, we have assumed that $s_1, s_2 > 0$. Indeed, it is possible to omit this assumption and obtain the associated LRCOMB as a consequence.

Remark 2. The interest of considering $s'_1 < s_1$ relies on the fact that we can relax the condition $\Lambda(H, T_j^1)$ *is even* for $s'_1 < j \leq s_1$. This case becomes specially interesting when $z_j \sim 1$ for $s'_1 < j \leq s_1$ because in such a case the strength of the comb inequality remains almost the same.

3.3.2.7 Location-routing generalized large multistar inequalities

We now introduce a new class of location-routing generalized large multistar inequalities that are valid for the two-index vehicle-flow formulation and that cannot be derived from inequalities (3.34). For given $I' \subset I, S \subseteq J$, and $j \notin S$, define $\eta(I', S, j) = x(S : j) + 1/2x(I' : \{j\}) + y(I' : \{j\})$. The following Location-routing generalized large multistar inequality (LRGLM) is valid for the two-index vehicle-flow formulation:

$$x((I \setminus I') \cup \overline{S} : S) + 2y(I \setminus I' : S) \geq \frac{2}{Q}\left(d(S) - b(I') + \sum_{j \notin S} d_j \eta(I', S, j)\right). \tag{3.54}$$

The validity of constraints (3.54) is a consequence of the following lemma and proposition.

Lemma 3.3.6. *Let $I' \subset I$, $S \subseteq J$. Let $W_{I'}$ be the set of customers that are served from facilities in I', and $T \subseteq \overline{S} \cap W_{I'}$. Then $x(E(S)) + 1/2x(I' : S) + y(I' : S) \leq |S| - \frac{1}{Q}(d(S \cup T) - b(I'))$.*

Proposition 3.3.7. *Constraint* (3.54) *is valid for the CLRP.*

Remark A stronger valid inequality can be obtained by replacing the right-hand side of constraint (3.54) by

$$\frac{2}{Q}\left(d(S) - \sum_{i \in I'} b_i z_i + \sum_{j \notin S} d_j \eta(I', S, j)\right). \tag{3.55}$$

3.3.2.8 Lifted cover inequalities

Lifted cover inequalities (LCI) can be useful when facilities have heterogeneous capacities. In such a case, the valid knapsack inequality $\sum_{i \in I} b_i z_i \geq d(J)$ can be used in

order to derive LCI. For details on LCI we refer to Gu et al. [70].

3.4 Separation Algorithms

In this section we describe the separation algorithms that we use to identify violated valid inequalities from the families introduced in Section 3.3. We begin by introducing a general cut lifting heuristic that takes advantage of the particular underlying structure of some inequalities, decomposing the separation problem into two easier subproblems that are solved sequentially. Then, we present the different separation algorithms for the separation of the inequalities presented in the paper. They include some exact separation algorithms based on maximum-flow computations as well as connected components or shrinking heuristics. We make use of the CONCORDE Library [34] to solve the maximum-flow problems as well as the connected components problems, and the COMBO algorithm [102] for solving 0-1 knapsack problems.

3.4.1 A cut lifting heuristic

In this section we describe a general separation algorithm that takes advantage of the special structure of some families of valid inequalities. Let us consider a polytope $\mathscr{X} = \{x \in \mathbb{R}_n, Ax \leq b\}$ and denote by $\mathscr{Y} = conv(\mathscr{X} \cap \mathbb{Z}_n)$ the convex hull of the integer points of $\mathscr{X}$. Given a function $f : \mathbb{R}_n \to \mathbb{R}$ and a scalar $g \in \mathbb{R}$, we say that the tuple (f,g) is a valid inequality for $\mathscr{Y}$ if $f(x) \leq g$ for every $x \in \mathscr{Y}$. Given two functions $f : \mathbb{R}_n \to \mathbb{R}$ and $h : \mathbb{R}_n \to \mathbb{R}$ let us denote by $[f+h]$ the function $[f+h](x) = f(x)+h(x)$. Suppose that we are given a family of valid inequalities for $\mathscr{Y}$, $\mathscr{F} = \{([\alpha_j + \beta_{jk}], \gamma_j) : j = 1, \dots \mathscr{J}, k = 1, \dots, \mathscr{K}_j\}$ with $\beta_{jk}(\cdot) \geq 0$ for all j,k. Suppose that the family $\mathscr{F}_1 = \{(\alpha_j, \gamma_j) : j = 1, \dots, \mathscr{J}\}$ is easy to separate, in the sense that for any $\varepsilon > 0$ and $x \in \mathscr{X}$ the decision problem

$$\exists j \in \{1, \dots, \mathscr{J}\} \text{ such that } \alpha_j(x) > \gamma_j - \varepsilon \qquad (P_1)$$

is easy to solve. Suppose that for given $j \in \{1, \ldots, \mathscr{J}\}$ and $x \in \mathscr{X}$, the problem

$$\begin{aligned} \max_{k} \quad & f(k) = \beta_{jk}(x) \\ \text{s.t.} \quad & k \in \mathscr{K}_j \end{aligned} \tag{P_2}$$

is easy to solve also, or that a good lower bound can be computed efficiently. Thus, given $x \in \mathscr{X}$, the following heuristic aims to find a valid cut $([\alpha_j + \beta_{jk}], \gamma_j) \in \mathscr{F}$ that is violated by x:

i. Fix $\varepsilon > 0$ and use separation procedures for problem (P_1) in order to find one or more j's such that $\alpha_j(x) > \gamma_j - \varepsilon$. We say that we find an ε-$\mathscr{F}_1$ cut.

ii. For every j found in (i) solve problem (P_2), obtaining k. If $\alpha_j(x) + \beta_{jk}(x) > \gamma_j$ then a violated inequality has been identified.

This procedure, although not exact, decomposes the problem into two easier subproblems and, as we will see later, can take advantage of known separation algorithms for related families of inequalities. We will see that problem (P_2) usually corresponds to solving a 0-1 knapsack problem. This problem is weakly $\mathscr{NP}$-hard and efficient exact algorithms have been proposed. We have chosen to use the COMBO algorithm [102] that stands as the state-of-the-art solver for the 0-1 knapsack problem.

3.4.2 CVRP Inequalities

For the CVRP inequalities we make use of the separation algorithms developed by Lysgaard et al. [101] and which are available at the author's website [100].

3.4.3 y-Capacity constraints

We use the cut lifting heuristic described in Section 3.4.1 to exploit the well-known separation algorithms for the capacity constraints of the CVRP. In fact, problem (P_1) corresponds to the separation of the CC. Suppose that a set S has been found that solves the ε-CC separation problem. Problem (P_2) then aims to find a subset $S' \subseteq S$ such that

the quantity $y(I : S')$ is maximum while respecting the constraint $r(S \setminus S') = r(S)$. This problem can be written as the following 0-1 knapsack problem:

$$\begin{aligned} \max_{\mu} \quad & \sum_{j \in S} \mu_j y(I : j) \\ \text{s.t.} \quad & \sum_{j \in S} d_j \mu_j \leq \begin{cases} d(S) - Q(r(S) - 1) - 1 & \text{if } d(S) \not\equiv 0 \pmod{Q} \\ 0 & \text{otherwise} \end{cases} \\ & \mu \in \{0,1\}^{|S|}. \end{aligned}$$

In our implementation, we have modified the code of Lysgaard et al. [101] to find ε-CC. The 0-1 knapsack problem is solved to optimality using the COMBO algorithm.

3.4.4 Strengthened facility capacity inequalities

We introduce separation algorithms for the separation of the SFCI (3.49). Note that as for the three-index case the inequalities (3.50) dominate (3.49), so the separation algorithms for the latter can in fact be safely used as heuristics. The separation for SFCI constraints (3.49) is done in three stages. First, we obtain candidate sets S and facilities $i \in I$ by solving the separation problem for the particular case of $|I'| = 1, |S'| = 0$. We refer to these specific constraints as the Basic FCI (BFCI). Note that these constraints are enough to ensure the feasibility of solutions. For each candidate sets S and $I' = \{i\}$, we use a greedy heuristic to enlarge the set I', and at every iteration in which I' is enlarged, we compute the set $S' \subset S$ that maximizes the quantity $y(I', S')$ and such that $r(S, I') = r(S \setminus S', I')$. This last problem corresponds to a 0-1 knapsack problem with item sizes $(d_j)_{j \in S}$, weights $(y(I', \{j\}))_{j \in S}$ and knapsack capacity of either $d(S) - b(I') - Q(r(S, I') - 1) - 1$ if $d(S) - b(I') \neq 0 \pmod{Q}$ or 0 otherwise. This procedure is an application of the cut lifting heuristic described in the subsection above, in which the subproblem corresponds to the described knapsack problem. We now describe the separation routine for generating the candidate sets S and $\{i\}$. We have implemented a safe shrinking routine, a connected component heuristic and an exact routine for the

fractional case based on a series of min-cut computations, all of which are applied in the following order:

i. Start applying the shrinking routine. Every time that two customers are chosen for shrinking, the shrinking heuristic is applied to these customers.

ii. If the shrinking process is completed and the shrinking routine is not able to find a violated BFCI we run a connected component heuristic over the connected components of the shrunk graph.

iii. If none of the above procedures is able to find a violated BFCI we solve a polynomial number of min-cut problems over the shrunk graph.

We now present in detail the safe shrinking routine as well as each of the heuristic procedures mentioned above.

3.4.4.1 A safe shrinking routine

In what follows we denote by ω^*, ϕ^* the weight functions obtained from x^* and y^*, respectively, after successive contractions, and we keep x^*, y^* for the weights in the original unshrunk graph. We denote by d^* the aggregated demands of super-customers as well, whose set we denote by J_S. A super-customer comprises the set of all customers that have been shrunk to the same super-node. We will show that it is safe to shrink two customers $h, j \in J_S$ whenever

i. $d^*_h + d^*_j \leq Q$ and

ii. $[\omega^*_{hj} \geq 1]$ or $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$.

Let us start by fixing a facility i. We will first show that for the separation of a BFCI using facility i it is safe to shrink any pair of nodes h and j in J_S satisfying only condition ii. If this is the case and $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$, the new weights for the shrunk node $\{h, j\}$ are

- $\omega^*_{\{h,j\}v} = 0$ for all $v \in I \cup (J_S \setminus \{h, j\})$
- $\phi^*_{\{h,j\}v} = \begin{cases} 1 & \text{if } v = i \\ 0 & \text{otherwise} \end{cases}$

Otherwise (i.e., if $\omega^*_{hj} \geq 1$), the new weights are recalculated using the usual rule, as follows:

- $\omega^*_{\{h,j\}v} = \omega^*_{hv} + \omega^*_{jv}$ for all $v \in I \cup (J_S \setminus \{h, j\})$.

Remark For every super-customer h in the shrunk graph it is true that $\omega^*(\delta(h)) + 2\phi^*(I : h) = 2$.

Lemma 3.4.1. *For fixed $i \in I$, it is safe to shrink nodes $h, j \in J_S$ such that $\omega^*_{hj} \geq 1$ or $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$.*

Remark If $\phi^*_{ih} = 1$ and $\phi^*_{ij} = 1$ it is not true that the shrinking of h and j is safe when considering a BFCI using a different facility, say l. In fact, in such a case, the last inequality in the proof above will be $\sigma_l(T) - \sigma_l(S) \leq 2 - 2\omega^*_{hj} - (\omega^*_{lh} + 2\phi^*_{lh})$ which is equal to 2. The next lemma proves, however, that in this case and whenever $d^*_j \leq Q$ and $d^*_h \leq Q$, h and j can be safely omitted from any BFCI containing facility l.

Lemma 3.4.2. *Let $h \in J_S$ be such that $\phi^*_{ih} = 1$ and $d^*_h \leq Q$. It is safe to omit node h from any BFCI containing a facility $l \neq i$.*

The following corollary follows as a consequence of Lemmas 3.4.1 and 3.4.2.

Corollary 3.4.3. *It is safe to shrink customers h, j such that*

i. $d^*_h + d^*_j \leq Q$ *and*

ii. $\omega^*_{hj} \geq 1$ *or* $\phi^*_{ih} = \phi^*_{ij} = 1$ *for some* $i \in I$.

3.4.4.2 Shrinking heuristic

During the execution of the shrinking routine, we can check at every iteration of the algorithm if two given super-customers $h, j \in J_S$ violate a BFCI, i.e., if $\omega^*_{hj} + \frac{1}{2}\omega^*(i : \{h, j\}) + \phi^*(i : \{h, j\}) > 2 - r(\{h, j\}, \{i\})$ for some $i \in I$. If this is the case, a violated inequality is obtained. Otherwise, we continue shrinking.

3.4.4.3 Connected component heuristic

Given a family of weights $(x'_e)_{e\in E}$, let $G_{x'} = (V, E_{x'})$ be the graph induced by the edges of E with strictly positive weights x'_e. The connected component heuristic works under the principle that if a violated BFCI exists associated to a facility i, then there is one contained in one of the connected components of the graph $G_{x'}$ (see Lemma 3.4.4 below), with x' defined as follows:

$$x'_e = \begin{cases} x^*_e + 2y^*_e & \text{if } e \in \delta(I) \\ x^*_e & \text{otherwise.} \end{cases} \tag{3.56}$$

Lemma 3.4.4. *Let $i \in I$ be a facility, and let $S \subseteq J$ be a disconnected (with respect to x') customer subset. Without loss of generality suppose that S_1, S_2 is a partition of S such that both S_1 and S_2 satisfy the CC constraints* (3.2)*. Then, if (i,S) defines a violated BFCI, (i,S_1) or (i,S_2) define another BFCI cut with a stronger violation as measured by the difference between the right-hand side and left-hand side of constraint* (3.49) *evaluated in vectors (x^*, y^*).*

The description of the algorithm is as follows. We start by looking at the connected components of the graph $G_{x'}$ (we make sure that connected components of $G_{x'}$ will satisfy constraints CC during their separation). Let S_k, I_k be the customers and facilities belonging to the k^{th} connected component, for $k = 1, \ldots, \Gamma$. Then, for every k and for every $i \in I_k$ we set $S^i_k = S_k \setminus \{h : y^*_{lh} = 1, l \neq i\}$, and we iteratively check whether the pair (i, S^i_k) violates a BFCI or not. If it does, we have identified a violated inequality. Otherwise we choose $j \in S^i_k$ such that the quantity $x^*(S^i_k \setminus \{j\} : j) + 1/2x^*_{ij} + y^*_{ij} + r(S^i_k, \{i\}) - r(S^i_k \setminus \{j\}, \{i\})$ is minimum and we remove it from S^i_k, repeating this procedure as long as we do not find a violated cut and $S^i_k \neq \emptyset$.

3.4.4.4 Exact separation of fractional BFCI's

The problem of finding a violated fractional BFCI can be formulated as the solution of $|I||J|$ minimum $\{s,t\}$-cut problems as follows: fix some $i \in I$ and $j \in J$. Consider the

graph $G'(V',A')$ produced from $G(V,A)$ after deleting node i and contracting nodes in $I \setminus \{i\}$ in a single super node s. Define the weight of the new edges $\{h,k\} \in A', h < k$ as

$$x'_{hk} = \begin{cases} \sum_{l \in I \setminus \{i\}} (x^*_{lk} + 2y^*_{lk}) - 2d_k/Q & \text{if } h = s \\ x^*_{hk} & \text{if } h \neq s. \end{cases}$$

Although there are negative weight edges, the problem of finding a minimum $\{s, j\}$-cut can still be solved in polynomial time as pointed out by McCormick et al. [103]. Obviously there exists an $s-j$ cut in the modified graph of capacity smaller than $-2b_i/Q$ for some $i \in I, j \in J$ iff there exists a violated fractional BFCI.

3.4.5 Effective strengthened facility capacity inequalities

Analougously to the SFCI, note that for the three-index case, the separation procedures for constraints (3.51) can be safely used as heuristics for separating constraints (3.52). The separation of the ESFCI (3.51) is done in a completely analogous way to the SFCI. In a first stage, we get candidate sets S and $I' = \{i\}$ by solving the separation algorithms for the EBFCI, that correspond to the particular case of ESFCI when $|I'| = 1, |S'| = 0$. For every candidate pair S, $I' = \{i\}$, we enlarge the set I' in a greedy way and after every extension we compute the set S' that maximizes the quantity $y(I',S')$ and such that $r(S,I') = r(S \setminus S',I'), r(S,I' \setminus \{i\}) = r(S \setminus S',I' \setminus \{i\})$. Again, this problem corresponds to a 0-1 knapsack problem and is a direct application of the cut lifting heuristic. The separation algorithms for the EBFCI are completely analogous to those used for the BFCI and for the sake of brevity we will omit the remaining details.

Remark Note that the safe shrinking result for the BFCI is also safe for the separation of the EBFCI. In fact, one can take advantage of this observation and shrink the graph just once.

3.4.6 Co-circuit constraints

We have implemented two heuristic procedures and an exact algorithm based on the computation of a minimum-cut tree. Note that for a given set S, the computation of the set F such that the left-hand side of constraint (3.38) is minimum can be done in linear time by defining $F = \{e \in \delta(S) : x_e \leq 1/2\}$. If $|F|$ is even, then we either add to or remove from F the edge in $\delta(S)$ that minimizes the increase of the left-hand side of (3.38). The first heuristics checks, for every customer $j \in S$ if the corresponding co-circuit constraint is violated for $S = \{j\}$. If we de not find any cut, we compute the blocks (2-connected components) of the graph $G_{1/2}$ induced by the edges $\{e \in E : \varepsilon \leq x_e \leq 1-\varepsilon\}$ and whose weights are taken as $w_e = \min\{x_e, 1-x_e\}$. For this, we have taken $\varepsilon = 10^{-5}$. If this procedure also fails, then we solve the separation of the blossom inequalities by computing a minimum-cut tree on graph $G_{1/2}$ using the Gomory-Hu algorithm [67]. We take as candidate handles the cuts induced by the edges of this tree. The first heuristic and the exact separation are done as suggested by Belenguer et al. [19], while the idea of considering the blocks of the graph as candidate handles has been successfully implemented into the separation of blossom inequalities in the CONCORDE solver for the TSP [5]. The separation of the DCoCC is done in a completely analogous way to the CoCC and, for the sake of brevity, we omit the details.

3.4.7 Facility degree constraints

Constraints (3.39) are not dynamically added but rather included at the beginning of the algorithm for the set J_Q built as follows. Let $J_Q = \emptyset$ and let V be the set containing the customers in J sorted by non-decreasing demands. Pick the first customer $v \in V$ and check if $d_v + d_j \leq Q$ for all $j \in J_Q$. If that is the case, then add v to J_Q, remove v from V and continue. If not, then stop. This way of constructing the set J_Q generalizes the approach of Belenguer et al. [19] in which J_Q is restricted to contain customers whose demands are $\leq Q/2$ by adding the possibility of adding one more customer.

For the separation of constraints (3.40) (respectively (3.48)) we have implemented two heuristics. First, we fix $i \in I$ and set $S = \emptyset$. Iteratively we enlarge set S by adding

the customer $j \notin S$ that maximizes the quantity $2y^*_{ij} + x^*_{ij} + x^*(S:j)$ (respectively $2y^*_{ij} + x^{i*}_{ij} + x^{i*}(S:j) - u_{ij}$). The algorithm terminates if either $d(S) \geq Q$ or a violated constraint (3.40) (respectively (3.48)) has been detected. If this fails, we check the violation for every y-CC generated so far during the algorithm such that $d(S) \leq Q$, just as done by Belenguer et al. [19].

3.4.8 Path constraints

To separate constraints (3.6) we first shrink the graph using a safe shrinking routine. Once the graph has been completely shrunk we find (if one exists) a violated constraint (3.6) using a greedy search heuristic or, in case the first fails, a series of min-cut computations, which yields an exact separation algorithm.

3.4.8.1 A safe shrinking routine

Using the same notation as before, let J_S be the customer set containig the shrunk customers, and let ω^*, ϕ^* be the edge weights in the shrunk graph. The following proposition gives a safe condition for shrinking customer nodes during the separation of constraints (3.6).

Proposition 3.4.5. *For the path constraints* (3.6) *it is safe to shrink customers* $u, v \in J_S$ *such that* $\omega^*_{uv} \geq 1$ *and* $\omega^*(I:u) = \omega^*(I:v) = 0$.

3.4.8.2 Greedy search heuristic

Because solving a max-flow problem can be time-consuming, we have implemented a greedy search heuristic that aims to find all the chains of length two or three in the shrunk graph. We simply check for every pair or triplet of customers (in the shrunk graph) whether they define or not a violated PC.

3.4.8.3 Exact separation

The problem of finding a violated inequality (3.6) can be solved in polynomial time [19] in the following way. For fixed $h, j \in J$ contract in the usual way (i.e., by recalculat-

ing the edge weights properly) in the underlying graph $G(V,E)$ nodes in I in a super-node s and nodes h, j in a super-node t. Let us call $J' = (J - \{h, j\}) \cup \{t\}$. In this new graph, let us consider the following weight function:

$$x'_{uv} = \begin{cases} x^*(I : v) & u = s, v \in J' \setminus \{t\} \\ x^*(I : \{h, j\}) & u = s, v = t \\ x^*_{uv} & u, v \in J' \setminus \{t\} \\ x^*_{uh} + x^*_{uj} & u \in J' \setminus \{t\}, v = t. \end{cases}$$

Let us find a cut of minimum weight between s and t in this graph. Let S be the side of this minimum cut that contains t. Then, define $I_1 = \emptyset$. For every $i \in I$, if $x^*_{ih} > x^*_{ij}$ then make $I_1 \leftarrow I_1 \cup \{i\}$. By construction, sets S, I_1 will violate constraint (3.6) iff they define a violated PC. As only a polynomial number of maximum-flow problems are solved, the method remains polynomial in $|I \cup J|$.

3.4.9 Location-routing comb inequalities

We present a tabu search algorithm for separating a subset of constraints LRCOMB in which $|T_j \cap I| \in \{0, 1\}$ for all j. Given a customer set H and t teeth $\Pi = (T_j)_{j=1}^t$ we call them a pseudo-comb if they satisfy conditions (iii)-(iv) of the definition of a comb, and $|T_j \cap I| \leq 1$ for all $1 \leq j \leq t$. Our separation algorithm proceeds in three stages: i) We search for ε-strengthened comb inequalities (SCI), getting candidate handles and teeth; ii) We use a greedy heuristic that breaks intersections (teeth can intersect in a SCI) by deleting elements that appear in two or more teeth from those that make the violation the greatest. If all the depots appear in a tooth, we delete all these depots except the one with the greatest violation. This process is repeated as many times as needed in order to obtain a pseudo-comb; iii) For every candidate pseudo-comb found after i) and ii), we proceed with the following tabu search metaheuristic.

Let us consider a pseudo-comb $C = (H, \Pi = (T_j)_{j=1}^t)$. Define $v(C)$ equal to the difference between the left-hand side of (3.53) and the right-hand side of (3.53). If C is

a valid comb, then $v(C)$ represents the violation of the comb. Let us define the pseudo-violation $\mu(C)$ equal to

$$\mu(C) = v(C) - \sum_{j=1}^{t} \delta(H \cap T_j = \emptyset) - \sum_{j=1}^{t} \delta(T_j \setminus H = \emptyset) \\ - \delta(\widehat{r}(H,\Pi) \equiv_2 0) - \sum_{j=1}^{t} \delta(\Lambda(H,T_j) \equiv_2 1).$$

The idea of considering the pseudo-violation instead of just the violation is justified by the fact that our procedure passes through pseudo-combs. Let $\mathscr{T}$ be the tabu list. A member l of $\mathscr{T}$ has two components, say $n(l)$ equal to a node and $pos(l)$ equal to a position relative to the comb. Here $pos(l)$ can take four values: $H \setminus \Pi, H \cap \Pi, \Pi \setminus H$ and $\overline{(H,\Pi)}$, where $\overline{(H,\Pi)}$ is the set containing all nodes not in the pseudo-comb (H,Π). Constructed in this way, the goal of the list $\mathscr{T}$ is to forbid the movement of a node $n(l)$ to position $pos(l)$ during a certain number of iterations.

Given a pseudo-comb $C = (H, \Pi = (T_j)_{j=1}^{t})$ we consider several simple neighborhoods, all of which can be evaluated very quickly.

$\mathbf{N}_1$ Pick a customer j from $H \setminus \Pi$ and remove it from C. Add $(j, H \setminus \Pi)$ to T.

$\mathbf{N}_2$ Pick a customer j from $H \cap \Pi$ and remove it from H. Add $(j, H \cap \Pi)$ to T.

$\mathbf{N}_3$ Pick a customer j from $H \cap \Pi$ and remove it from Π. Add $(j, H \cap \Pi)$ to T.

$\mathbf{N}_4$ Pick a customer j from $\Pi \setminus H$ and remove it from C. Add $(j, \Pi \setminus H)$ to T.

$\mathbf{N}_5$ Pick a facility i from $\Pi \setminus H$ and remove it from C. Add $(i, \Pi \setminus H)$ to T.

$\mathbf{N}_6$ Pick a customer j from $\overline{C}$ and add it to $H \setminus \Pi$. Add $(j, \overline{C})$ to T.

$\mathbf{N}_7$ Pick a customer j from $\Pi \setminus H$ and add it to H. Add $(j, \Pi \setminus H)$ to T.

$\mathbf{N}_8$ Pick a customer j from $H \setminus \Pi$ and add it to Π. Add $(j, H \setminus \Pi)$ to T.

$\mathbf{N}_9$ Pick a customer j from $\overline{C}$ and add it to $\Pi \setminus H$. Add $(j, \overline{C})$ to T.

$\mathbf{N}_{10}$ Pick a facility i from $\overline{C}$ and add it to $\Pi \setminus H$. Add $(i, \overline{C})$ to T.

The neighborhoods are sorted in such a way that removal and insertion movements are alternated. If, after inspecting some neighborhood, we get a pseudo-violation of

value greater than the incumbent, we update the incumbent and restart. Otherwise, we continue with the next neighborhood. We have found convenient to start the next iteration inspecting the first neighborhood not inspected during the last iteration. If we finish inspecting all the neighborhoods without finding any pseudo-comb with value greater than the incumbent, we update it with the best movement found and restart. During the process we do not consider movements of nodes to a tabu position, thus decreasing the probability of cycling. Note also that for neighborhoods N_5 and N_{10}, the contribution to the pseudo-violation depends on whether we are in the case $1 \leq j \leq s_1$ or $s_1 < j \leq s_2$ in the definition of a comb. We have chosen to make this distinction by simply considering the value of z_i in the current iteration. In fact, if $z_i < 0.75$ we consider the first case, otherwise the second. The algorithm finishes when we have found a valid comb with positive pseudo-violation, or when a maximum number of iterations has been performed without success. In our experiments we have noticed that most combs were found during the first 30 iterations. We have thus set the maximum number of iterations to 300 for the root node and 50 for the remaining nodes.

3.4.10 y-Generalized large multistar inequalities

The separation problem for the y-GLM (3.26) can be done in polynomial time by solving a maximum $\{s,t\}$-flow problem in the following graph $G' = (V', E')$. Let s and t be two dummy nodes, and let $V' = J \cup \{s,t\}$, $E' = E(J) \cup \{\{s,j\} : j \in J\} \cup \{\{j,t\} : j \in J\}$. With every edge $e \in E'$ we associate a capacity x'_e defined by

$$x'_e = \begin{cases} x^*(I:\{j\}) + 2\frac{d_j}{Q}\left(y^*(I:\{j\}) - 1\right) & e = \{s,j\}, j \in J \\ 0 & e = \{j,t\}, j \in J \\ x^*_e\left(1 - 2\frac{d_j}{Q}\right) & e = \{h,j\}, h,j \in J. \end{cases} \tag{3.57}$$

It is easy to check that a maximum $\{s,t\}$-flow exists in this graph with negative value iff there is a violated y-GLM. However, note that while maximum-flow algorithms assume positive edge capacities, this may not happen. Indeed, if $2d_j \leq Q$ for all $j \in J$ then the usual weight transformation on the edges joined to nodes s or t suffices.

Suppose, however, that for some $j \in J$, $2d_j > Q$. The following transformation proposed by Blasum and Hochstättler [22] can be applied in order to get a non-negative weight digraph whose minimum-cut coincides with the one we are looking for. Define for every $j \in J$ the quantities $\underline{d}_j = \min\{\frac{Q}{2}, d_j\}, \overline{d}_j = d_j - \underline{d}_j$. Let us consider the following weight function:

$$x'_e = \begin{cases} x^*(I:\{j\}) + 2\frac{d_j}{Q}\left(y^*(I:\{j\}) - 1\right) & e = \{s,j\}, j \in J \\ -2\sum_{v \in J}\left(\frac{d_v}{Q} - \frac{d_j}{Q}\right)x^*_{jv} & e = \{j,t\}, j \in J \\ x^*_{hj}\left(1 - 2\left(\frac{d_h}{Q} + \frac{\overline{d}_j}{Q}\right)\right) & e = \{h,j\}, h, j \in J. \end{cases} \tag{3.58}$$

It can be checked that a maximum $\{s,t\}$-flow in this modified graph is also a maximum flow in the original graph, and thus the separation algorithm of the y-GLM is polynomially solvable.

3.4.11 y-Location-routing generalized large multistar inequalities

The separation of constraints (3.34) is done in two stages. In the first stage, we separate what we call the Basic y-LRGLM (B-y-LRGLM) that corresponds to a y-LRGLM in the particular case of $|H| = 1$. For every $H = \{i\} \subset I$ we run an exact polynomial-time algorithm based on a maximum-flow computation, obtaining a candidate set S. Then, we use a greedy algorithm for enlarging the set H by inserting at each iteration the facility that makes the violation the greatest. For the separation of the B-y-LRGLM let us fix a facility $i \in I$, and let us consider two dummy nodes s,t. Let us consider the following graph $G_i = (V_i, E_i)$, with $V_i = J \cup \{s,t\}$, $E_i = \delta(J) \cup (\{s\} : J) \cup (J : \{t\}))$, weighted as follows:

$$x'_e = \begin{cases} x^{I\setminus\{i\}}(I:j) + \frac{2d_j}{Q}\left(\sum_{l \in I\setminus\{i\}}(y_{lj} - u_{lj})\right) & e = \{s,j\}, j \in J \\ 0 & e = \{j,t\}, j \in J \\ x^{I\setminus\{i\}}_{hj}\left(1 - \frac{2d_j}{Q}\right) & h, j \in J. \end{cases} \tag{3.59}$$

Again, if the weights on the edges are negative, we apply the same transformation as

for the separation of the y-GLM. It is easy to check that a violated B-y-LRGLM exists iff the minimum $\{s,t\}$-cut in this graph is negative.

3.4.12 Location-routing generalized large multistar inequalities

We have implemented the following heuristic procedure for the separation of the LRGLM (3.54) strengthened using as right-hand side the expression (3.55). First, we use an exact algorithm for finding a ε-LRGLM in the particular case in which $|I'| = 1$. We call these inequalities Basic LRGLM (B-LRGLM). For every pair of sets S and $I' = \{i\}$ found by this procedure, we apply a greedy heuristic that iteratively enlarges I' and checks for the violation of the corresponding LRGLM. The exact procedure used for the separation of the B-LRGLM is as follows.

Let $i \in I$, and let us consider a digraph whose vertex set is $J \cup \{i\} \cup \{s\}$, where s is the node obtained by the contraction of facilities in $I \setminus \{i\}$. The edge set is determined by the non-zero weights in the arcs, given by

$$x'_{uv} = \begin{cases} -\frac{d_j}{Q}(x^*_{iu} + 2y^*_{iu}) & u \in J, v = i \\ x^*(I \setminus \{i\} : u) + 2y^*(I \setminus \{i\} : u) - \frac{2d_u}{Q} & u = s, v \in J \\ x^*_{uv}\left(1 - 2\frac{d_v}{Q}\right) & u, v \in J. \end{cases}$$

It is easy to check that a violated LRGLM exists iff a minimum $\{s,i\}$-cut in this digraph has value less than $-\frac{2b_i}{Q}z^*_i$. In the case of negative weights, we apply the same procedure already described for the separation of the y-GLM. Thus, the problem of finding a LRGLM can be solved in polynomial time by computing a minimum $\{s,i\}$-cut in this graph.

3.4.13 Lifted cover inequalities

We add LCI only at the root node. We use the algorithm of Gu et al. [70] for finding violated LCI. For details on the algorithm we refer to Gu et al. [70].

3.5 The exact algorithms

We test the models, separation routines and valid inequalities introduced in this paper by developing four branch-and-cut algorithms. The first, named VFF2, is a branch-and-cut over the two-index vehicle-flow formulation (VF2) augmented by the valid inequalities introduced in this paper except those that are specific to the three-index formulations. The second algorithm, named VFF3, is a branch-and-cut on the three-index vehicle-flow formulation augmented by all the valid inequalities. The third algorithm, named CFF2, is branch-and-cut algorithm over the two-index two-commodity flow formulation (CF2) augmented by all the inequalities introduced in this paper except those that are specific to the three-index formulations and the y-GLM. The fourth algorithm, named CFF3, is a branch-and-cut algorithm over the three-index two-commodity flow formulation (CF3) augmented by all the inequalities except for y-GLM, LRGLM and y-LRGLM. For the two-commodity formulations, we also replace vehicle-flow variables x with their corresponding commodity-flow variables w using identities (3.22) and (3.30) and by adding (as cutting planes) inequalities (3.44) and (3.46) for the two-index and three-index formulations commodity-flow formulations, respectively. For the vehicle-flow formulations, we also add inequalities (3.43) and (3.45) dynamically as cutting planes.

3.5.1 The separation strategies

The separation strategies for the different formulations depend on two criteria: strength of the inequalities and need for feasibility. Inequalities that are needed to impose feasibility of integer solutions are thus separated first, while the rest are added as cutting planes, and among these two families the priority is given to inequalities that, in our experiments, have shown a bigger impact on the lower bounds. The exception are inequalities FAI that are separated immediately after the LCI. After some preliminary tests we have found the convenience of separating these inequalities before any other family of cuts. In addition, inequalities ESFCI and SFCI that seem to have an important impact in formulations VFF3, CFF2 and CFF3. Although they are not needed to impose feasibility, they are also separated first. Taking these observations into account, we have

decided to divide the inequalities into two groups: those that are statically separated (i.e., separated in every node of the branching tree) and those for which we dynamically decide whether to separate them or not in a certain node of the branching tree. The criteria for selecting these dynamic cuts are explained later. In Table 3.I we describe the two groups of inequalities as well as their separation order for each of the four different formulations considered in our study.

Form.	Static Cuts	Dynamic Cuts
VFF2	FAI (3.43), y-CC, ESFCI (3.51), SFCI (3.49), SPC	LCI, FDC (3.40), CoCC, FrCI, y-GLM, LRGLM, SCI, LRCOMB, MSI, HYP
VFF3	FAI (3.45), y-CC, ESFCI (3.52), SFCI (3.50), y-GLM, y-LRGLM, LRGLM	LCI, FDC (3.40) and (3.48), CoCC, DCoCC, SPC, FrCI, SCI, LRCOMB, MSI, HYP
CFF2	FAI (3.44), y-CC, ESFCI (3.51), SFCI (3.49), SPC	LCI, FDC (3.40), CoCC, FrCI, LRGLM SCI, LRCOMB, MSI, HYP
CFF3	FAI (3.46), y-CC, ESFCI (3.52), SFCI (3.50)	LCI, CoCC, DCoCC, FDC (3.40) and (3.48), SPC, FrCI, SCI, LRCOMB, MSI, HYP

Table 3.I: Separation order of valid inequalities

Note that in order to avoid errors due to floating point arithmetic, a certain tolerance $\varepsilon > 0$ must be imposed for checking the violation of a certain cut. Moreover, if ε is too small, many cuts whose violations are very close to zero will be added without much impact on the lower bound. After a series of experiments, we have decided to use $\varepsilon = 0.1$ for all of the cuts except for hypotour inequalities and multistar inequalities for which the tolerance was set to $\varepsilon = 0.4$. At the root node, all families of cuts are separated. Moreover, all separation algorithms are used for each family. More specifically, for the FrCI, the tree size is set to a maximum of 10,000 nodes, as for the LRCOMB the number of iterations of the Tabu Search is set to 300.

For the cutting strategy in nodes other than the root, we use the following approach. For each family of dynamic cuts (see Table 3.I), say for family $\mathscr{C}$, we let $n(\mathscr{C})$ be the number of times that a cut of family $\mathscr{C}$ has been found to be violated and thus added to the problem. We keep track of this quantity in the different branches of the tree and at certain depths we check whether $\mathscr{C}$ has been useful in the current branch. If $n(\mathscr{C}) = 0$

then the family $\mathscr{C}$ is not separated anymore during the current branch. For the other cuts, say those such that $n(\mathscr{C}) > 0$ the counter is reset to 0. After some testing we have decided to perform this check for the first time at depth 10 and then for multiples of 5. In practice, we have verified that no dynamic cuts are present after depth 25. Note also that the tree size in the separation of the inequalities FrCI is lowered to 200 nodes, while the maximum number of iterations of the Tabu Search algorithm for the separation of LRCOMB is lowered to 50.

Regarding the setting of the cut lifting heuristic described in Section 3.4.1, we have performed a series of tests in order to choose the value of ε that fits best with every cut family. The values that we have tested are ε equal to 0, 0.25, 0.50 and 1.0. For y-CC we have decided to set $\varepsilon = 0.25$ at the root node and $\varepsilon = 0$ for the remaining nodes (recall that cuts are not added unless they are violated by more than 0.1). For the SFCI and ESFCI we have set $\varepsilon = 0.25$ during the whole computation.

3.5.2 The branching strategy

We use the following branching strategy. We first branch on location variables z. If no variable z is fractional, we branch on cutsets. For this, we use and idea proposed in Belenguer et al. [19]: during the root relaxation, each y-CC cut is added as an equality constraint by adding an extra slack variable to the problem. We then let CPLEX branch on these slack variables. For the three-index formulations, we then branch on the assignment variables u. Finally, we branch on the vehicle-flow variables y or x (or in their equivalent expressions using variables w in the commodity-flow formulations). We have observed that while strong branching produces the smallest branching trees, the computational effort is too high and for hard instances it is not worthwhile. On the other hand, branching on the most fractional variables leads to much bigger branching trees. Thus, we let CPLEX branch based on pseudo costs, which we found to give the best balance between lower bound quality and CPU time.

3.6 Computational experience

In this section we describe the implementation of the algorithms as well as the results obtained on a series of instances from the litterature.

The algorithms have been coded in C++ using the Concert Technology framework of CPLEX 12.2. Tests were run on an Intel Xeon E5462, 3.0 Ghz processor with 16GB of memory under the Linux Operating System kernel 2.6. In order to obtain results purely related to the strength of the formulations and the cuts used in this paper, other families of cuts added by CPLEX (such as MIR, knapsack cover, GUB, clique, etc.) have been disabled. Finally, the node selection strategy has been set to best-first search.

We have run our algorithm on four datasets taken from the literature. The instances descriptions are as follows:

i. Set $\mathscr{S}_1$ contains 17 instances adapted by Barreto [18] from other problems in the literaure. Only three instances have facilities with limited capacities.

ii. Set $\mathscr{S}_2$ contains 24 randomly generated instances from the experiments of Belenguer et al. [19]. All of the instances have facilities with limited capacities. Customer loads are taken randomly in the interval $[11,20]$ and capacities are set in such a way that: 1) the average number of customers served by a vehicle is either 5 or 10, and 2) two or three facilities are required for serving the whole demand. Note that no customer with extremely low (10 units or less) or extremely high (more than 20 units) demands is present.

iii. Set $\mathscr{S}_3$ contains 12 randomly generated instances from the experiments of Akca et al. [4]. Facilities all have limited capacities, chosen in such a way that at least two of the facilities must be open. The vehicle capacities are such that the average number of customers per route is between 4 and 7, and that the longest route serves at most 8 customers.

iv. Set $\mathscr{S}_4$ contains 6 instances with capacitated vehicles and uncapacitated facilities from the experiments of Tuzun and Burke [142]. The fixed costs of the facilities are relatively low compared to the routing costs.

Additionally, sets $\mathscr{S}_1$ and $\mathscr{S}_2$ are also subdivided into small instances (those with 50 customers or less) and large instances (those having more than 50 customers). We have used the upper bounds reported by Baldacci et al. [16] as cutoff values during the branch-and-bound search. The idea is to measure the efficiency of each of the formulations for closing the optimality gap. For a self-contained methodology, these upper bounds should be obtained by a suitable heuristic, which is beyond the scope of this paper. The data sets can all be obtained from the website `http://www.crt.umontreal.ca/~ccontard`. We have designed and implemented five sets of experiments.

In the first set of experiments, we compute the linear relaxation lower bound for each of the four formulations, and compare their quality as well as the CPU time taken by each of them. The results are reported in Tables 3.II-3.V. In these tables, columns labeled z^* represent the upper bound for each instance. Columns labeled *gap (%)* and *t (s)* stand for the relative gap (for a given lower bound z_{lb} it is computed as $(z^* - z_{lb})/z^* \times 100$) and the CPU time in seconds. As shown by these tables, algorithms VFF3 and CFF3 normally produce the tightest lower bounds, at the expense of much larger computing times. However, algorithms VFF2 and CFF2 are the fastest to compute their respective lower bounds. There are two possible readings for these results. On the one hand, compact two-index formulations give reasonably good lower bounds in very short computing times. Therefore, much larger branching trees can be inspected during the same amount of time, with respect to formulations with more variables. On the other hand, the lower bounds obtained by three-index formulations are in some cases much tighter than the ones obtained by the two-index formulations. Therefore, the structure of the CLRP is better captured in the former case, and in some instances the differences are dramatic (like on instances of set $\mathscr{S}_4$). One could thus ask, whether it is possible or not to tighten two-index formulations with valid inequalities so to produce lower bounds that are comparable to those obtained by three-index formulations.

In the second set of experiments, we have run the four algorithms for a maximum time of two hours. The objective is to test and compare their efficiency to rapidly solve some relatively easy instances. In Tables 3.VI-3.IX, columns are similar to previous tables. We have added a column labeled *nodes* that reports the number of nodes inspected

during the branching tree. As we can see, formulation VFF2 gives the best results on average. Indeed, it is able to solve 32 instances, four more than VFF3, three more than CFF2 and 6 more than CFF3. However, three-index formulations produce tighter gaps on instances ppw-50x5-0b and ppw-50x5-2b. This suggests that two-index formulations are not able in those cases to capture some important underlying information of the CLRP structure that is indeed beneficial to three-index formulations. Moreover, instance ppw-50x5-0b is solved to optimality only by formulation VFF3. The overall conclusion is that compact two-index formulations produce the best average results at the expense of underestimating some important information.

In the third set of experiments, we have run the algorithms for a maximum time of 12 hours. The objective is to measure the efficiency of each formulations for solving of some hard instances of the CLRP. The results are summarized in Tables 3.X-3.XIII. The columns are the same as for the previous experiments. Now, the number of instances solved is 32 for VFF2 and VFF3, 30 for CFF2 and 29 for CFF3. Note that this increase in the cpu time has a marginal impact on the performance of two-index formulations, whereas three-index formulations seem to scale better. The is due mainly to the fact that branching has a lower impact on trees of large size, which is typically the case with compact two-index formulations.

In the fourth set of experiments, we compare algorithm VFF2 against the branch-and-cut algorithm of Belenguer et al. [19]. In Table 3.XIV, headers *# instances*, *# solved* and *avg. gap* stand for the total number of instances, the number of instances solved to optimality and the average optimality gap on each subset of instances. Our implementation of the branch-and-cut algorithm VFF2 is able to produce tighter gaps in average than the one of Belenguer et al. [19]. Moreover, our algorithm scales to solve some large instances with up to 100 customers (ppw-100x10-2b, P112212, P113212), while the method of Belenguer et al. [19] was able to solve instances with up to 50 customers. Several refinements in our implementation might explain these results, including the use of stronger inequalities, efficient separation algorithms, as well as the dynamic separation strategy during the branching tree that deactivates cuts that do not seem promising in a certain branch.

In the fifth and last set of experiments, we compare the results obtained by our branch-and-cut algorithms against the branch-and-cut-and-price method of Baldacci et al. [16]. In Table 3.XV we summarize the number of instances solved by their method (column BMW) against the instances solved by all of our methods (column FF). As shown in this table, the branch-and-cut-and-price method of Baldacci et al. [16] is able to solve much more instances than all of the flow formulations together. This is not a surprising result since column generation algorithms are based on much tighter formulations. However, it is worth noticing that their method failed to solve instance ppw-50x5-2b which has been solved by algorithm VFF3, and also solves instance ppw-50x5-0b in a much longer time than VFF3, which suggests that some of the inequalities introduced in this paper would deserve being included into set-partitioning formulations.

3.7 Concluding Remarks

We have introduced three new flow formulations for the CLRP that dominate, in terms of the linear relaxation lower bound, the previous two-index vehicle-flow formulation of Belenguer et al. [19]. We derive new valid inequalities for each of the formulations and strengthen some of the previously known inequalities. In addition, we are able to obtain new classes of multistar inequalities for the vehicle-flow formulations as linear combinations of the degree constraints and assignment constraints for the commodity-flow formulations. For each of the inequalities used in this paper, we introduce separation algorithms that are either new or that generalize the separation methods introduced by Belenguer et al. [19]. We have implemented suitable branch-and-cut algorithms using each of the three formulations introduced in this paper plus the original two-index vehicle-flow formulation and present computational results comparing them. The results show that, in most cases, compact formulations produce the tightest gaps in the long run due to their ability to perform more branching nodes. However, on some hard instances where facility capacities are important, three-index formulations seem to be the right choice (like on instances ppw-50x5-0b, ppw-50x5-2b, ppw-100x5-3b, ppw-100x10-3b). This is a direct consequence of an important drawback of compact two-index formula-

tions with respect to three-index formulations, and it is the fact that it is not possible to follow the flow leaving from a facility at every single node of the graph. We also compare the algorithms used in this paper against the state-of-the-art solvers for solving the CLRP, namely the branch-and-cut method of Belenguer et al. [19] and the branch-and-cut-and-price of Baldacci et al. [16]. The results show that our implementation of the branch-and-cut on the two-index vehicle-flow formulation produces tighter gaps than the one of Belenguer et al. [19], and is able to scale and solve large instances with up to 100 customers. The branch-and-cut-and-price algorithm of Baldacci et al. [16] in general outperforms the flow-based algorithms; however, it is worth remarking that on two instances (ppw-50x5-0b, ppw-50x5-2b) the three-index formulation obtained tighter gaps, and even solved ppw-50x5-2b which no other exact method did before. These results suggest that taking into consideration the facilities from where the flow originates has significant impact on the performance of an exact algorithm. As an avenue of future research, we believe that embedding some of the inequalities introduced in this paper into a branch-and-cut-and-price algorithm could result in a more robust exact algorithm for the CLRP.

Acknowledgements

The authors would like to thank the *Natural Sciences and Engineering Research Council of Canada* (NSERC) and *Le fonds québécois de la recherche sur la nature et les technologies* (FQRNT) for their financial support.

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
Perl83-12x2	204.00	0.61	0.01	0.00	0.03	0.48	0.08	0.00	0.03
Gas67-21x5	424.90	3.99	0.21	3.12	0.84	4.12	0.44	2.98	0.53
Gas67-22x5	585.11	0.10	0.04	0.10	0.24	0.10	0.07	0.10	0.41
Min92-27x5	3062.02	5.62	0.29	2.15	2.26	6.24	0.39	2.64	1.32
Gas67-29x5	512.10	4.89	0.46	3.26	2.11	4.76	1.90	3.64	1.74
Gas67-32x5	562.22	5.72	0.51	3.90	3.59	5.72	1.26	4.05	1.33
Gas67-32x5-2	504.33	3.27	0.80	1.86	2.17	3.24	1.25	2.19	1.29
Gas67-36x5	460.37	1.30	1.40	1.16	8.28	1.35	5.35	0.71	9.42
Chr69-50x5ba	565.62	5.62	3.74	4.41	14.32	5.63	6.40	3.83	6.76
Chr69-50x5be	565.60	8.85	3.04	7.34	16.15	8.82	15.44	5.90	5.45
Perl83-55x15	1112.06	3.42	6.17	2.44	676.90	3.42	14.29	2.06	64.25
Chr69-75x10ba	886.30	14.47	23.54	11.67	1406.12	14.63	164.87	11.47	306.63
Chr69-75x10be	848.85	10.42	15.25	7.77	1920.82	9.84	227.24	7.40	284.03
Chr69-75x10bmw	802.08	9.27	20.18	6.68	1165.53	9.69	92.79	6.58	237.99
Perl83-85x7	1622.50	2.53	18.93	2.06	966.41	2.53	199.44	1.96	153.35
Das95-88x8	355.78	5.73	13.15	4.81	1028.53	6.03	66.88	4.73	336.98
Chr69-100x10	833.43	4.89	9.71	4.07	2299.52	4.82	51.37	3.79	709.63
Average		5.34	6.91	3.93	559.64	5.38	49.97	3.77	124.77
Average in small instances		4.00	1.05	2.73	5.00	4.05	3.26	2.60	2.83
Average in large instances		7.25	15.28	5.64	1351.98	7.28	116.70	5.43	298.98

Table 3.II: Gaps and CPU times after linear relaxation on instances of set $\mathscr{S}_1$

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
ppw-20x5-0a	54793	4.57	0.35	3.74	1.00	4.54	0.83	3.89	0.68
ppw-20x5-0b	39104	0.00	0.02	0.00	0.10	0.00	0.04	0.00	0.14
ppw-20x5-2a	48908	2.71	0.26	2.31	0.59	2.78	0.76	2.35	0.53
ppw-20x5-2b	37542	0.00	0.01	0.00	0.06	0.00	0.02	0.00	0.10
ppw-50x5-0a	90111	10.94	27.10	5.98	100.66	10.89	51.78	5.88	26.38
ppw-50x5-0b	63242	7.50	5.05	6.64	28.54	7.76	14.70	6.67	16.58
ppw-50x5-2a	88298	7.52	5.08	5.81	36.84	7.50	17.22	5.84	11.04
ppw-50x5-2b	67308†	5.63	2.75	5.74	16.20	5.66	12.78	5.72	16.38
ppw-50x5-2a'	84055	1.95	29.50	1.89	65.33	1.97	125.81	1.93	25.09
ppw-50x5-2b'	51822	0.86	1.76	0.72	19.48	0.85	3.51	0.82	9.82
ppw-50x5-3a	86203	10.23	14.67	5.15	72.97	10.20	52.76	5.26	25.99
ppw-50x5-3b	61830	6.26	4.38	5.30	23.36	5.84	9.94	5.22	9.67
ppw-100x5-0a	274814	3.56	2509.03	2.82	4955.67	3.59	3117.51	2.86	1218.73
ppw-100x5-0b	214392	3.21	391.48	3.09	5605.32	3.18	1508.11	3.33	10298.00
ppw-100x5-2a	193671	3.77	365.93	2.17	2402.35	3.81	3127.55	2.21	460.93
ppw-100x5-2b	157173	2.34	83.27	1.91	816.75	2.32	613.30	1.96	365.99
ppw-100x5-3a	200079	8.82	108.07	2.23	2331.79	8.81	1664.74	2.39	441.84
ppw-100x5-3b	152441	5.08	27.40	2.62	792.25	5.08	148.33	2.66	163.83
ppw-100x10-0a	289018	7.88	1133.84	5.78	7281.82	7.34	4708.86	4.97	1249.72
ppw-100x10-0b	234641	4.74	147.20	4.53	4060.38	4.78	1235.21	4.45	1638.18
ppw-100x10-2a	243590	4.07	1473.84	3.28	7285.58	4.11	3823.62	3.21	1214.73
ppw-100x10-2b	203988	2.50	90.42	2.48	1928.96	2.46	612.71	2.34	1308.84
ppw-100x10-3a	252421	8.65	740.38	6.17	7228.90	8.72	3433.17	6.28	1188.19
ppw-100x10-3b	204597	5.00	112.22	4.75	4384.28	4.99	1011.74	4.60	857.56
Average		4.91	303.08	3.55	2059.97	4.88	1053.96	3.53	856.21
Average in small instances		4.85	7.58	3.61	30.43	4.83	24.18	3.63	11.87
Average in large instances		4.97	598.59	3.49	4089.50	4.93	2083.74	3.44	1700.55

† New upper bound found.

Table 3.III: Gaps and CPU times after linear relaxation on instances of set $\mathscr{S}_2$

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
cr30x5a-1	819.5	3.33	0.89	2.06	3.71	3.29	1.62	2.99	2.09
cr30x5a-2	821.5	5.89	0.41	5.29	2.61	5.90	0.97	4.92	1.67
cr30x5a-3	702.3	0.56	0.71	0.09	2.52	0.73	1.22	0.38	2.66
cr30x5b-1	880.0	7.39	0.52	5.91	3.00	7.35	1.55	5.61	1.35
cr30x5b-2	825.3	3.52	1.31	1.65	3.71	3.62	3.31	1.72	1.54
cr30x5b-3	884.6	3.33	1.09	2.14	4.47	3.25	2.73	2.20	2.01
cr40x5a-1	928.1	8.95	1.32	8.01	9.20	8.96	3.28	7.08	2.01
cr40x5a-2	888.4	8.83	1.04	6.17	9.91	8.92	1.95	6.09	3.63
cr40x5a-3	947.3	7.47	2.48	6.31	9.43	7.45	7.56	5.50	4.91
cr40x5b-1	1052.0	10.26	2.80	6.64	16.68	10.13	6.70	6.52	4.66
cr40x5b-2	981.5	8.57	1.26	3.70	16.18	8.42	4.55	3.79	5.41
cr40x5b-3	964.3	4.51	2.32	2.92	17.40	4.46	8.22	2.94	3.94
Average		6.05	1.35	4.24	8.23	6.04	3.64	4.14	2.99

Table 3.IV: Gaps and CPU times after linear relaxation on instances of set $\mathscr{S}_3$

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
P111112	1467.69	12.64	16.31	7.94	2180.38	12.60	119.93	6.91	313.17
P111212	1394.8	15.92	41.04	11.37	1763.18	15.91	214.59	9.09	451.88
P112112	1167.16	11.69	42.24	3.69	3245.86	11.91	339.71	3.72	547.29
P112212	791.66	19.99	53.95	2.97	1021.34	20.01	111.13	2.94	274.73
P113112	1245.45	19.27	31.51	7.84	4987.70	19.51	130.71	7.74	637.84
P113212	902.26	16.49	83.95	1.82	4383.76	16.90	774.21	1.96	1601.76
Average		16.00	44.83	5.94	2930.37	16.14	281.71	5.39	637.78

Table 3.V: Gaps and CPU times after linear relaxation on instances of set $\mathscr{S}_4$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
Perl83-12x2	204.00	0.00	0.02	0	0.00	0.05	0	0.00	0.11	3	0.00	0.08	0
Gas67-21x5	424.90	0.00	0.59	26	0.00	4.29	19	0.00	1.91	39	0.00	4.76	20
Gas67-22x5	585.11	0.00	0.07	0	0.00	0.45	1	0.00	0.19	7	0.00	1.41	9
Min92-27x5	3062.02	0.00	0.73	5	0.00	3.94	3	0.00	1.78	20	0.00	7.87	6
Gas67-29x5	512.10	0.00	1.01	10	0.00	7.28	13	0.00	3.84	20	0.00	18.06	11
Gas67-32x5	562.22	0.00	1.76	47	0.00	19.90	39	0.00	4.98	39	0.00	27.58	47
Gas67-32x5-2	504.33	0.00	1.01	2	0.00	4.72	5	0.00	2.32	7	0.00	3.96	9
Gas67-36x5	460.37	0.00	2.80	4	0.00	15.50	7	0.00	12.07	16	0.00	43.64	23
Chr69-50x5ba	565.62	0.00	44.78	224	0.00	530.77	169	0.00	212.91	384	0.00	1655.58	314
Chr69-50x5be	565.60	0.00	68.79	232	0.00	1093.98	351	0.00	200.49	258	0.00	4999.94	476
Perl83-55x15	1112.06	0.70	7496.92	2755	1.45	7269.41	30	0.85	7193.60	1090	1.14	7215.43	98
Chr69-75x10ba	886.30	9.03	7408.62	1869	10.67	7401.99	16	9.07	7193.13	1304	10.38	7315.95	22
Chr69-75x10be	848.85	3.26	7367.52	2394	6.10	7339.88	18	3.64	7193.51	1293	5.76	7472.10	28
Chr69-75x10bmw	802.08	3.37	7468.83	1766	5.25	7351.64	22	3.69	7194.08	1403	5.16	7417.08	30
Perl83-85x7	1622.50	0.68	7557.62	1404	1.29	7259.75	14	0.75	7193.95	1044	1.17	7250.88	19
Das95-88x8	355.78	0.00†	411.15	129	0.73	7267.20	8	0.00†	1308.87	289	0.32	7275.53	21
Chr69-100x10	833.43	0.51	7385.58	1777	2.83	7326.33	9	0.80	7193.80	330	2.46	7397.48	17
Average		1.03	2659.87	744	1.67	3111.59	43	1.11	2641.86	444	1.55	3418.08	68
Instances solved		11			10			11			10		
Average in small instances		0.00	12.16	55	0.00	168.09	61	0.00	44.06	79	0.00	676.29	92
Small instances solved		10			10			10			10		
Average in large instances		2.51	6442.32	1728	4.05	7316.60	17	2.69	6352.99	965	3.77	7334.92	34
Large instances solved		1			0			1			0		

† Optimality proven for the first time.

Table 3.VI: Gaps and CPU times after 2 hours on instances of set $\mathscr{S}_1$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
ppw-20x5-0a	54793	0.00	5.04	704	0.00	12.61	2	0.00	21.70	1009	0.00	25.59	197
ppw-20x5-0b	39104	0.00	0.03	0	0.00	0.16	0	0.00	0.11	8	0.00	0.50	6
ppw-20x5-2a	48908	0.00	1.31	137	0.00	5.93	1	0.00	4.65	191	0.00	7.44	77
ppw-20x5-2b	37542	0.00	0.02	0	0.00	0.12	0	0.00	0.08	9	0.00	0.49	3
ppw-50x5-0a	90111	1.77	7336.51	9778	2.46	7207.13	186	2.39	7193.53	5286	2.78	7210.67	949
ppw-50x5-0b	63242	1.23	7304.54	16772	0.00	1865.40	59	1.77	7193.39	6014	0.38	7199.65	1608
ppw-50x5-2a	88298	1.00	7265.57	13181	1.05	7198.80	193	1.21	7193.73	8548	1.24	7197.10	1990
ppw-50x5-2b	67308	1.17	7282.73	17630	0.68	7199.72	114	1.50	7193.41	7854	0.87	7201.81	1086
ppw-50x5-2a'	84055	0.28	7244.32	24306	0.48	7203.79	183	0.49	7193.36	11205	0.55	7203.78	1949
ppw-50x5-2b'	51822	0.00	10.64	134	0.00	136.19	7	0.00	41.93	289	0.00	125.11	91
ppw-50x5-3a	86203	1.16	7283.89	4915	2.09	7199.29	113	1.46	7194.24	2894	2.20	7210.56	632
ppw-50x5-3b	61830	0.00	71.76	596	0.00	647.18	17	0.00	378.75	1000	0.00	1368.30	348
ppw-100x5-0a	274814	2.36	7293.80	350	2.82	7454.50	185	2.48	7196.30	119	2.68	7515.34	77
ppw-100x5-0b	214392	2.19	7420.00	460	3.09	7438.56	111	2.41	7197.83	197	3.33	10298.00	0
ppw-100x5-2a	193671	1.60	7398.86	726	2.17	7413.93	118	1.65	7196.96	505	2.21	7281.76	0
ppw-100x5-2b	157173	0.78	7323.10	1777	1.40	7270.06	97	0.88	7195.16	531	1.17	7231.19	12
ppw-100x5-3a	200079	1.44	7340.95	1379	1.93	7316.05	364	1.64	7197.17	546	1.64	7270.58	290
ppw-100x5-3b	152441	0.57	7332.99	1225	0.74	7240.37	204	0.49	7196.00	329	0.85	7272.68	25
ppw-100x10-0a	289018	3.74	7394.33	59	5.78	7281.82	114	7.02	7243.67	0	4.49	7455.56	369
ppw-100x10-0b	234641	2.48	7334.20	821	4.50	7509.37	68	2.98	7194.80	32	4.17	7919.87	5
ppw-100x10-2a	243590	1.40	7351.82	764	3.28	7285.58	95	1.54	7197.52	73	2.10	7505.96	330
ppw-100x10-2b	203988	0.00	4734.83	8710	0.97	7293.31	24	0.11	7193.51	1698	0.82	7410.55	30
ppw-100x10-3a	252421	4.02	7326.18	85	6.17	7228.90	100	4.70	7195.95	16	4.98	7459.89	579
ppw-100x10-3b	204597	2.15	7338.32	1033	4.00	7412.70	55	2.55	7194.91	56	3.95	7382.64	4
Average		1.22	5391.49	4398	1.82	5284.23	100	1.55	5417.03	2017	1.68	5698.13	444
Instances solved		7			7			6			6		
Average on small instances		0.55	3650.53	7346	0.56	3223.03	73	0.73	3634.07	3692	0.67	3729.25	745
Small instances solved		6			7			6			6		
Average on large instances		1.89	7132.45	1449	3.07	7345.43	128	2.37	7199.98	342	2.70	7667.00	143
Large instances solved		1			0			0			0		

Table 3.VII: Gaps and CPU times after 2 hours on instances of set $\mathscr{S}_2$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
cr30x5a-1	819.5	0.00	3.23	67	0.00	27.25	11	0.00	7.65	88	0.00	35.05	46
cr30x5a-2	821.5	0.00	8.77	663	0.00	44.50	253	0.00	59.83	1244	0.00	38.20	234
cr30x5a-3	702.3	0.00	0.91	0	0.00	3.50	0	0.00	1.77	38	0.00	7.82	27
cr30x5b-1	880.0	0.00	9.05	395	0.00	86.25	235	0.00	34.66	439	0.00	90.66	184
cr30x5b-2	825.3	0.00	2.55	8	0.00	14.78	9	0.00	7.26	113	0.00	14.57	43
cr30x5b-3	884.6	0.00	3.25	40	0.00	21.85	11	0.00	11.06	137	0.00	28.52	92
cr40x5a-1	928.1	0.00	140.31	1308	0.00	1391.66	783	0.00	608.40	1342	0.14	7328.59	2143
cr40x5a-2	888.4	0.00	86.31	1655	0.00	591.68	601	0.00	203.61	935	0.00	1059.93	692
cr40x5a-3	947.3	0.00	76.63	1019	0.00	785.90	463	0.00	409.66	1286	0.00	1494.26	542
cr40x5b-1	1052.0	0.00	3115.92	21471	0.86	7197.27	2396	0.57	7193.77	9358	1.03	7196.37	2243
cr40x5b-2	981.5	0.00	7.61	25	0.00	104.38	45	0.00	46.77	1011	0.00	91.69	85
cr40x5b-3	964.3	0.00	12.33	20	0.00	50.08	31	0.00	24.30	74	0.00	54.01	49
Average		0.00	288.91	2223	0.07	859.93	403	0.05	717.39	1339	0.10	1453.31	532
Instances solved		12			11			11			10		

Table 3.VIII: Gaps and CPU times after 2 hours on instances of set $\mathscr{S}_3$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
P111112	1467.69	2.76	7529.07	427	5.69	7389.51	9	3.17	7197.69	125	5.01	7361.46	25
P111212	1394.8	2.86	7444.58	594	8.39	7341.34	10	3.99	7193.81	147	6.42	7471.62	28
P112112	1167.16	0.49	7362.85	2205	2.22	7331.45	10	0.72	7197.28	542	2.22	7533.47	7
P112212	791.66	0.00	4980.13	8747	0.98	7326.49	10	0.20	7194.37	2763	0.97	7365.17	7
P113112	1245.45	3.64	7312.07	1954	7.83	7417.84	2	3.86	7195.71	537	4.99	7598.02	13
P113212	902.26	0.00	247.09	33	0.15	7373.92	3	0.00	1634.29	139	0.13	7689.31	5
Average		1.63	5812.63	2327	4.21	7363.43	7	1.99	6268.86	709	3.29	7503.18	14
Instances solved		2			0			1			0		

Table 3.IX: Gaps and CPU times after 2 hours on instances of set $\mathscr{S}_4$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
Perl83-12x2	204.00	0.00	0.02	0.00	0.00	0.05	0.00	0.00	0.11	3.00	0.00	0.08	0
Gas67-21x5	424.90	0.00	0.59	26.00	0.00	4.29	19.00	0.00	1.91	39.00	0.00	4.76	20
Gas67-22x5	585.11	0.00	0.07	0.00	0.00	0.45	1.00	0.00	0.19	7.00	0.00	1.41	9
Min92-27x5	3062.02	0.00	0.73	5.00	0.00	3.94	3.00	0.00	1.78	20.00	0.00	7.87	6
Gas67-29x5	512.10	0.00	1.01	10.00	0.00	7.28	13.00	0.00	3.84	20.00	0.00	18.06	11
Gas67-32x5	562.22	0.00	1.76	47.00	0.00	19.90	39.00	0.00	4.98	39.00	0.00	27.58	47
Gas67-32x5-2	504.33	0.00	1.01	2.00	0.00	4.72	5.00	0.00	2.32	7.00	0.00	3.96	9
Gas67-36x5	460.37	0.00	2.80	4.00	0.00	15.50	7.00	0.00	12.07	16.00	0.00	43.64	23
Chr69-50x5ba	565.62	0.00	44.78	224.00	0.00	530.77	169.00	0.00	212.91	384.00	0.00	1655.58	314
Chr69-50x5be	565.60	0.00	68.79	232.00	0.00	1093.98	351.00	0.00	200.49	258.00	0.00	4999.94	476
Perl83-55x15	1112.06	0.45	44278.60	14429.00	1.09	43228.20	85.00	0.57	43163.30	8651.00	0.77	43188.20	430
Chr69-75x10ba	886.30	8.53	43923.00	9619.00	10.31	43345.80	30.00	8.55	43162.10	7889.00	9.37	43289.40	452
Chr69-75x10be	848.85	2.69	43902.30	12090.00	5.58	43353.70	33.00	2.95	43161.20	7070.00	4.44	43355.40	102
Chr69-75x10bmw	802.08	2.91	44528.40	8496.00	4.44	43315.00	66.00	3.20	43166.10	8238.00	4.12	43409.60	189
Perl83-85x7	1622.50	0.52	44470.70	6905.00	0.84	43224.30	98.00	0.59	43164.30	6428.00	0.87	43215.60	93
Das95-88x8	355.78	0.00	411.15	129.00	0.00	26533.80	99.00	0.00	1308.87	289.00	0.00	25008.10	160
Chr69-100x10	833.43	0.26	43933.90	9206.00	1.51	43396.80	61.00	0.47	43168.30	1891.00	1.72	43376.40	52
Average		0.90	15621.74	3613	1.40	16945.79	63	0.96	15337.34	2426	1.25	17153.27	141
Instances solved		11			11			11			11		
Average in small instances		0.00	12.16	55	0.00	168.09	61	0.00	44.06	79	0.00	676.29	92
Small instances solved		10			10			10			10		
Average in large instances		2.19	37921.15	8696	3.39	40913.94	67	2.33	37184.88	5779	3.04	40691.81	211
Large instances solved		1			1			1			1		

Table 3.X: Gaps and CPU times after 12 hours on instances of set $\mathscr{S}_1$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
ppw-20x5-0a	54793	0.00	5.04	704	0.00	12.61	107	0.00	21.70	1009	0.00	25.59	197
ppw-20x5-0b	39104	0.00	0.03	0	0.00	0.16	0	0.00	0.11	8	0.00	0.50	6
ppw-20x5-2a	48908	0.00	1.31	137	0.00	5.93	113	0.00	4.65	191	0.00	7.44	77
ppw-20x5-2b	37542	0.00	0.02	0	0.00	0.12	0	0.00	0.08	9	0.00	0.49	3
ppw-50x5-0a	90111	1.37	43595.50	33712	2.05	43176.50	4476	1.97	43167.70	18081	2.33	43180.30	4192
ppw-50x5-0b	63242	0.98	43595.50	58868	0.00	1865.40	2605	1.57	43167.20	20673	0.00	13194.70	4362
ppw-50x5-2a	88298	0.75	43418.50	47481	0.80	43172.10	7203	0.97	43167.60	31819	0.91	43170.40	8336
ppw-50x5-2b	67308	0.84	43574.60	61257	0.00[†]	42721.30	14967	1.19	43167.60	27182	0.30	43175.20	4949
ppw-50x5-2a'	84055	0.14	43350.80	95114	0.33	43177.50	11615	0.31	43167.20	50204	0.35	43175.00	10497
ppw-50x5-2b'	51822	0.00	10.64	134	0.00	136.19	155	0.00	41.93	289	0.00	125.11	91
ppw-50x5-3a	86203	0.94	43479.80	18603	1.71	43199.40	2784	1.26	43167.90	10656	1.86	43177.50	2496
ppw-50x5-3b	61830	0.00	71.76	596	0.00	647.18	247	0.00	378.75	1000	0.00	1384.13	348
ppw-100x5-0a	274814	2.19	43556.70	6805	2.39	43510.20	260	2.28	43171.20	2850	2.58	43421.20	651
ppw-100x5-0b	214392	1.99	44111.80	4588	3.07	44111.10	3	2.18	43168.90	3699	3.21	43200.00	0
ppw-100x5-2a	193671	1.43	44427.70	4532	1.78	43425.80	277	1.48	43170.50	4004	2.02	43276.10	342
ppw-100x5-2b	157173	0.63	43711.50	9989	0.69	43224.60	1365	0.74	43165.80	3986	0.72	43206.70	544
ppw-100x5-3a	200079	1.32	43846.10	6623	1.49	43356.30	933	1.46	43167.30	8919	1.46	43259.60	4015
ppw-100x5-3b	152441	0.40	43541.60	7460	0.18	43205.30	2766	0.29	43165.90	3141	0.76	43225.00	19
ppw-100x10-0a	289018	3.36	43811.60	2120	4.25	43436.60	863	3.32	43171.80	692	4.36	43477.00	4820
ppw-100x10-0b	234641	2.27	43671.20	7250	3.51	43439.10	37	2.41	43168.70	1291	3.39	43836.90	506
ppw-100x10-2a	243590	1.29	43969.70	6861	1.85	43437.00	8	1.31	43170.60	2354	1.98	43475.00	4218
ppw-100x10-2b	203988	0.00	4734.83	8710	0.65	44250.50	2598	0.01	43176.70	13392	0.62	43436.80	3613
ppw-100x10-3a	252421	3.59	44012.80	1924	4.62	43456.40	2021	2.38	43171.00	684	4.83	43413.40	5648
ppw-100x10-3b	204597	1.94	43661.60	6506	1.39	44132.90	9	2.10	43168.30	928	2.90	43357.90	1155
Average		1.06	31173.36	16249	1.28	30879.17	2309	1.14	32395.38	8628	1.44	31300.08	2545
Instances solved		7			8			6			7		
Average on small instances		0.42	21758.62	26384	0.41	18176.20	3689	0.61	21621.03	13427	0.48	19218.03	2963
Small instances solved		6			8			6			7		
Average on large instances		1.70	40588.09	6114	2.16	43582.15	928	1.66	43169.72	3828	2.40	43382.13	2128
Large instances solved		1			0			0			0		

[†] Optimality proven for the first time.

Table 3.XI: Gaps and CPU times after 12 hours on instances of set $\mathscr{S}_2$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
cr30x5a-1	819.52	0.00	3.23	67	0.00	27.25	11	0.00	7.65	88	0.00	35.05	46
cr30x5a-2	821.50	0.00	8.77	663	0.00	44.5	253	0.01	59.83	1244	0.00	38.20	234
cr30x5a-3	702.30	0.00	0.91	0	0.00	3.5	0	0.00	1.77	38	0.00	7.82	27
cr30x5b-1	880.02	0.00	9.05	395	0.00	86.25	235	0.00	34.66	439	0.00	90.66	184
cr30x5b-2	825.32	0.00	2.55	8	0.00	14.78	9	0.00	7.26	113	0.00	14.57	43
cr30x5b-3	884.60	0.00	3.25	40	0.00	21.85	11	0.00	11.06	137	0.00	28.52	92
cr40x5a-1	928.10	0.00	140.31	1308	0.00	1391.66	783	0.00	608.40	1342	0.00	8619.65	2143
cr40x5a-2	888.42	0.00	86.31	1655	0.00	591.68	601	0.00	203.61	935	0.00	1059.93	692
cr40x5a-3	947.30	0.00	76.63	1019	0.00	785.9	463	0.00	409.66	1286	0.00	1494.26	542
cr40x5b-1	1052.04	0.00	3115.92	21471	0.00	27430.7	12839	0.00	15757.00	24688	0.00	38343.40	14040
cr40x5b-2	981.54	0.00	7.61	25	0.00	104.38	45	0.00	46.77	1011	0.00	91.69	85
cr40x5b-3	964.33	0.00	12.33	20	0.00	50.08	31	0.00	24.30	74	0.00	54.01	49
Average		0.00	288.91	22223	0.00	2546.04	1273	0.00	1431.00	2616.25	0.00	4156.48	1515
Instances solved		12			12			12			12		

Table 3.XII: Gaps and CPU times after 12 hours on instances of set $\mathscr{S}_3$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
P111112	14676.9	2.30	44319.80	3604	4.16	43338.20	37	2.49	43177.00	1535	4.65	43290.90	34
P111212	13948	2.20	44120.90	4199	5.08	43293.90	44	2.56	43167.00	1199	4.56	43392.10	66
P112112	11671.6	0.20	43745.40	13454	1.34	43449.90	23	0.29	43168.60	3934	0.81	43532.30	121
P112212	7916.6	0.00	4980.13	8747	0.23	43269.20	1035	0.05	43169.10	16394	0.33	43339.20	363
P113112	12454.5	3.45	43574.00	8057	4.14	43415.10	19	3.69	43167.00	3276	3.38	43576.00	135
P113212	9022.6	0.00	247.09	33	0.00	8247.08	45	0.00	1634.29	139	0.00	13705.40	206
Average		1.36	30164.55	6349	2.49	37502.23	2001	1.51	36247.17	4413	2.29	38472.65	154
Instances solved		2			1			1			1		

Table 3.XIII: Gaps and CPU times after 12 hours on instances of set $\mathscr{S}_4$

Family	# instances	BBPPW		VFF2	
		# solved	avg. gap	# solved	avg. gap
$\mathscr{S}_1$ small	10	8	0.00	10	0.00[†]
$\mathscr{S}_1$ large	7	0	2.84	1	1.63[†]
$\mathscr{S}_2$ small	12	6	0.70	6	0.55
$\mathscr{S}_2$ large[‡]	12	0	–	1	1.89
$\mathscr{S}_3$ all	12	12	0.00	12	0.00
$\mathscr{S}_4$ all[‡]	6	0	–	2	1.63
Total	59	26		32	

[†] Including only instances reported also in Belenguer et al. [19].

[‡] Instances not reported in Belenguer et al. [19].

Table 3.XIV: Overall results comparison on branch-and-cut algorithms

Family	# instances	BMW	FF
		# solved	# solved
$\mathscr{S}_1$ small	10	10	10
$\mathscr{S}_1$ large	7	5	1
$\mathscr{S}_2$ small	12	12	8
$\mathscr{S}_2$ large	12	6	1
$\mathscr{S}_3$ all	12	12	12
$\mathscr{S}_4$ all	6	5	2
Total	59	50	34

Table 3.XV: Overall results comparison against method of Baldacci et al. [16]

3.8 Proofs of lemmas and propositions

Proof of Proposition 3.2.1 It is direct to check that inequalities (3.24)-(3.25) imply the following inequalities

$$Qw_{ji} \leq (Q-d_j)(w_{ij}+w_{ji}) \qquad \{i,j\} \in \overline{E} \tag{3.60}$$

$$Qw_{ij} \geq d_j(w_{ij}+w_{ji}) \qquad \{i,j\} \in \overline{E}. \tag{3.61}$$

By adding identities (3.19) for customers $j \in S$ and after reducing we obtain

$$w(\delta^-(S)) + 2\sum_{j\in S} d_j y(I:\{j\}) = w(\delta^+(S)) + 2d(S)$$

By adding $w(\delta^+(S))$ at both sides of the identity above and after using identities (3.22) we obtain at the left-hand side $Qx(\delta(S)) + 2\sum_{j\in S} d_j y(I:\{j\})$. The desired right-

hand side is obtained after using constraint (3.61) for $w(\delta^+(S))$. □

Proof of Proposition 3.2.2 It is easy to see that the (DFI) imply the following inequalities

$$Qw^i_{jh} \leq (Q-d_j)(w^i_{hj}+w^i_{jh}) \qquad i \in I, \{h,j\} \in \overline{E} \tag{3.62}$$

$$Qw^i_{hj} \geq d_j(w^i_{hj}+w^i_{jh}) \qquad i \in I, \{h,j\} \in \overline{E} \tag{3.63}$$

By adding the flow conservation equations (3.27) for customers $j \in S$ and facilities $i \in I \setminus H$ we obtain

$$w^{I\setminus H}(\delta^-(S)) + 2\sum_{j\in S} d_j y(I\setminus H : \{j\}) = w^{I\setminus H}(\delta^+(S)) + 2\sum_{i\in I\setminus H}\sum_{j\in S} d_j u_{ij}$$

By adding $w^{I\setminus H}(\delta^+(S))$ at both sides of the above identity its left-hand turns to be equal to $Qx^{I\setminus H}(\delta(S)) + 2\sum_{j\in S} d_j y(I\setminus H : \{j\})$. For the right-hand size, we make use of the inequalities (3.62)-(3.63) in order to get $w^{I\setminus H}(\delta^+(S)) \geq \sum_{\substack{h\in S\\ j\notin S}} d_j x^{I\setminus H}_{hj}$. □

Proof of Proposition 3.3.1 if $x^i(F) < |F|$ then the constraint is trivially satisfied. If $x^i(F) = |F|$, then all the edges of F are used by vehicles linked to facility i. Since $|F|$ is odd, it follows that at least one edge, also linked to facility i, must be used in $\delta(S)\setminus F$. □

Proof of Proposition 3.3.2 if $\sum_{j\in S} u_{ij} = t$, then exactly t customers in S are served from facility i. For those customers, say S', given that $d(S') \leq Q$, and given that the triangular inequality holds between distances, then the customers in S' must be served all by the same vehicle. Indeed, if more than one vehicle serves S', then it is always possible to serve them at lower cost by a single vehicle. □

Proof of Proposition 3.3.3 If $y(I\setminus I' : S') = y(I\setminus I' : S'') = |S''|$ then $x((I\setminus I')\cup\overline{S} : S) =$

$x((I \setminus I') \cup (\overline{S \setminus S''}) : S \setminus S'')$, and then

$$\begin{aligned} x((I \setminus I') \cup \overline{S} : S) + 2y(I \setminus I' : S \setminus S') &= x((I \setminus I') \cup (\overline{S \setminus S''}) : S \setminus S'') + 2y(I \setminus I' : S \setminus S'') \\ &\geq r(S \setminus S'', I') \\ &\geq r(S \setminus S', I') \\ &= r(S, I'). \end{aligned}$$

□

Proof of Proposition 3.3.4 Let $S'' \subseteq S'$ such that $y(I \setminus I' : S') = y(I \setminus I' : S'') = |S''|$. This means that customer set S'' is served by single vehicles from $I \setminus I'$. Thus, $x((I \setminus I') \cup \overline{S} : S) = x((I \setminus I') \cup (\overline{S \setminus S''}) : S \setminus S'')$ so

$$\begin{aligned} x((I \setminus I') \cup \overline{S} : S) + 2y(I \setminus I' : S \setminus S') &= x((I \setminus I') \cup (\overline{S \setminus S''}) : S \setminus S'') + 2y(I \setminus I' : S \setminus S'') \\ &\geq 2r(S \setminus S'', I' \setminus \{i\}) + 2z_i(r(S \setminus S'', I') - r(S \setminus S'', I' \setminus \{i\})) \\ &\geq 2r(S \setminus S', I' \setminus \{i\}) + 2z_i(r(S \setminus S', I') - r(S \setminus S', I' \setminus \{i\})) \\ &= 2r(S, I' \setminus \{i\}) + 2z_i(r(S, I') - r(S, I' \setminus \{i\})). \end{aligned}$$

□

Proof of Proposition 3.3.5 Let us consider the SFCI and ESFCI in their weaker version that does not consider the subsets S'. These constraints can be written using the degree constraints as $x(E(S)) + \frac{1}{2}x(I' : S) + y(I' : S) \leq |S| - r(S, I')$ (for the SFCI) and $x(E(S)) + \frac{1}{2}x(I' : S) + y(I' : S) \leq |S| - r(S, I' \setminus \{i\}) + z_i(r(S, I' \setminus \{i\}) - r(S, I'))$ (for the ESFCI). We have

$$\begin{aligned} 2\alpha\bar{x} &\leq \sum_{u \in H} \bar{x}(\delta(u)) + \sum_{k=1}^{2} \sum_{j=1}^{s_k} (\bar{x}(E(T_j^k)) + \bar{x}(E(T_j^k \setminus H)) + \bar{x}(E(T_j^k \cap H))) \\ &\leq 2|H| + \sum_{k=1}^{2} \sum_{j=1}^{s_k} (\bar{x}(E(T_j^k)) + \bar{x}(E(T_j^k \setminus H)) + \bar{x}(E(T_j^k \cap H))). \end{aligned}$$

We now use the ESFCI in their inner form for $1 \leq j \leq s'_1$:

$$\begin{aligned}
\bar{x}(E(T_j^1)) \leq & \frac{1}{2}x(I_j : S_j^1) + y(I_j : S_j^1) + |S_j^1| - r(S_j^1, I_j \setminus \{i_j\}) \\
& + z_{i_j}(r(S_j^1, I_j \setminus \{i_j\}) - r(S_j^1, I_j)) \\
\leq & \frac{1}{2}x(I_j : J) + y(I_j : J) + |S_j^1| - r(S_j^1, I_j \setminus \{i_j\}) \\
& + z_{i_j}(r(S_j^1, I_j \setminus \{i_j\}) - r(S_j^1, I_j)) \\
\bar{x}(E(T_j^1 \setminus H)) \leq & \frac{1}{2}x(I_j : S_j^1 \setminus H) + y(I_j : S_j^1 \setminus H) + |S_j^1 \setminus H| - r(S_j^1 \setminus H, I_j \setminus \{i_j\}) \\
& + z_{i_j}(r(S_j^1 \setminus H, I_j \setminus \{i_j\}) - r(S_j^1 \setminus H, I_j)) \\
\leq & \frac{1}{2}x(I_j : J) + y(I_j : J) + |S_j^1 \setminus H| - r(S_j^1 \setminus H, I_j \setminus \{i_j\}) \\
& + z_{i_j}(r(S_j^1 \setminus H, I_j \setminus \{i_j\}) - r(S_j^1 \setminus H, I_j)) \\
\bar{x}(E(T_j^1 \cap H)) \leq & |S_j^1 \cap H| - r(S_j^1 \cap H)
\end{aligned}$$

and then

$$\begin{aligned}
\bar{x}(E(T_j^1)) + \bar{x}(E(T_j^1 \setminus H)) + \bar{x}(E(T_j^1 \cap H)) & \\
\leq x(I_j : J) + 2y(I_j : J) + 2|S_j^1| + z_{i_j}\Lambda(H, T_j^1) - \widehat{r}(H, T_j^1). &
\end{aligned}$$

For $s_1' < j \leq s_1$ we do a similar development obtaining

$$\bar{x}(E(T_j^1)) + \bar{x}(E(T_j^1 \setminus H)) + \bar{x}(E(T_j^1 \cap H)) \leq x(I_j : J) + 2y(I_j : J) + 2|S_j^1| - \widehat{r}(H, T_j^1).$$

For the remaining teeth we have

$$\bar{x}(E(T_j^2)) + \bar{x}(E(T_j^2 \setminus H)) + \bar{x}(E(T_j^2 \cap H)) \leq 2|S_j^2| - \widehat{r}(H, T_j^2).$$

Then, adding all these terms and bounding we obtain

$$2\alpha\bar{x} \leq 2|H| + \sum_{1\leq j\leq s_1} \big(x(I_j:J) + 2y(I_j:J)\big) + \sum_{1\leq j\leq s'_1} z_{i_j}\Lambda(H,T_j^1) + 2\sum_{k=1,2}\sum_{1\leq j\leq s_k} |S_j^k| - \widehat{r}(H,\Pi).$$

As $x(I_j:J)+2y(I_j:J)$ is *even* for $1\leq j\leq s_1$, $\Lambda(H,T_j^1)$ is *even* for $1\leq j\leq s'_1$ and $\widehat{r}(H,\Pi)$ is *odd*, after dividing by 2 and rounding the result follows. □

Proof of Lemma 3.3.6 If $S\subseteq W_{I'}$ then $d(S\cup T)\leq b(I')$ and the result is implied by the SFCI. If $S\subseteq \overline{W}_{I'}$ then $x(E(S))+\frac{1}{2}x(I':S)+y(I':S)\leq |S|-\frac{1}{Q}d(S)\leq |S|-\frac{1}{Q}(d(S\cup T)-b(I'))$. If $S=S_1\cup S_2, S_1=S\cap W_{I'}, S_2=S\cap\overline{W}_{I'}$, then $x(E(S))+x(I':S)+y(I':S)=\sum_{i=1,2}x(E(S_i))+\frac{1}{2}x(I':S_i)+y(I':S_i)\leq |S_1|+|S_2|-\frac{1}{Q}(d(S_1\cup T)-b(I')+d(S_2))$. □

Proof of Proposition 3.3.7 First, note that constraint (3.54) can be written, using the degree constraints, in the following equivalent form:

$$x(E(S))+\tfrac{1}{2}x(I':S)+y(I':S)+\tfrac{1}{Q}\sum_{j\notin S} d_j\eta(I',S,j)\leq |S|-\tfrac{1}{Q}(d(S)-b(I')). \tag{3.64}$$

Let us decompose the set $\overline{S}$ into three subsets $\overline{S}_0=\{j\in\overline{S}:\eta(I',S,j)=0\}$, $\overline{S}_{1/2}=\{j\in\overline{S}:\eta(I',S,j)=1/2\}$ and $\overline{S}_{1+}=\{j\in\overline{S}:\eta(I',S,j)\geq 1\}$. Using this for the summation in the left-hand side of the equation (3.64) we have

$$\sum_{j\notin S} d_j\eta(I',S,j)=\frac{1}{2}d(\overline{S}_{1/2})+\sum_{j\in\overline{S}_{1+}} d_j\eta(I',S,j). \tag{3.65}$$

But now, the second term of this last expression can be decomposed and bounded above as follows:

$$\begin{aligned}\sum_{j\in\overline{S}_{1+}} d_j\eta(I',S,j) &= \sum_{j\in\overline{S}_{1+}}(d_j-Q)\eta(I',S,j)+Q\sum_{j\in\overline{S}_{1+}}\eta(I',S,j)\\ &\leq d(\overline{S}_{1+})-Q|\overline{S}_{1+}|+Q\sum_{j\in\overline{S}_{1+}}\eta(I',S,j).\end{aligned}$$

Thus, the left-hand side of constraint (3.64) can be bounded above by

$$x(E(S)) + \tfrac{1}{2}x(I',S) + y(I',S) + \tfrac{1}{2Q}d(\overline{S}_{1/2}) + \tfrac{1}{Q}d(\overline{S}_{1+}) - |\overline{S}_{1+}| + \sum_{j\in\overline{S}_{1+}} \eta(I',S,j). \quad (3.66)$$

But now, we have

$$x(E(S)) + \tfrac{1}{2}x(I',S) + y(I',S) + \sum_{j\in\overline{S}_{1+}} \eta(I',S,j) \\ \leq x(E(S\cup\overline{S}_{1+})) + \tfrac{1}{2}x(I',S\cup\overline{S}_{1+}) + y(I',S\cup\overline{S}_{1+}).$$

Using this, (3.66) can be bounded above by

$$x(E(S\cup\overline{S}_{1+})) + \tfrac{1}{2}x(I',S\cup\overline{S}_{1+}) + y(I',S\cup\overline{S}_{1+}) + \tfrac{1}{2Q}d(\overline{S}_{1/2}) + \tfrac{1}{Q}d(\overline{S}_{1+}) - |\overline{S}_{1+}|.$$

Now, as $\overline{S}_{1/2} \subseteq W_{I'}$ we can apply the lemma and thus this last expression can be bounded above by

$$|S\cup\overline{S}_{1+}| - \tfrac{1}{Q}(d(S\cup\overline{S}_{1+}\cup\overline{S}_{1/2}) - b(I')) + \tfrac{1}{2Q}d(\overline{S}_{1/2}) + \tfrac{1}{Q}d(\overline{S}_{1+}) - |\overline{S}_{1+}| \\ \leq |S| - \tfrac{1}{Q}(d(S) - b(I')).$$

□

Proof of Lemma 3.4.1 Let $h, j \in J_S$ be such that $\omega^*_{hj} \geq 1$ or $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$. Let $S \subseteq J_S$ be a customer set crossing $\{h, j\}$, i.e., $S \cap \{h, j\}, S \setminus \{h, j\}$ and $\{h, j\} \setminus S \neq \emptyset$. Without loss of generality we suppose that $j \in S, h \notin S$. We will show that $T = S \cup \{h\}$ produces a violation of value at least that of S. Let us define $\sigma_i(T) = \omega^*((I \setminus \{i\}) \cup \overline{T} : T) + 2\phi^*(I \setminus \{i\} : T)$. Because $r(S, \{i\}) \leq r(T, \{i\})$ it suffices to show that $\sigma_i(T) \leq \sigma_i(S)$.

In fact

$$\begin{aligned}\sigma_i(T)-\sigma_i(S) =&[\omega^*(\delta(T))+2\phi^*(I:T)]-[\omega^*(\delta(S))+2\phi^*(I:S)]\\ &+[\omega^*(i:S)-\omega^*(i:T)]+2[\phi^*(i:S)-\phi(i:T)]\\ =&[\omega^*(\delta(T))+2\phi^*(I:T)]-[\omega^*(\delta(S))+2\phi^*(I:S)]-[\omega^*_{ih}+2\phi^*_{ih}].\end{aligned}$$

The submodularity of the cut function implies

$$\begin{aligned}&[\omega^*(\delta(T))+2\phi^*(I:T)]-[\omega^*(\delta(S))+2\phi^*(I:S)]\leq\\ &\qquad[\omega^*(\delta(\{h,j\}))+2\phi^*(I:\{h,j\})]-[\omega^*(\delta(j))+2\phi^*(I:j)]\end{aligned}$$

and then

$$\begin{aligned}\sigma_i(T)-\sigma_i(S)&\leq\omega^*(\delta(h))+2\phi^*(I:h)-2\omega^*_{hj}-(\omega^*_{ih}+2\phi^*_{ih})\\ &\leq 2-2\omega^*_{hj}-(\omega^*_{ih}+2\phi^*_{ih}).\end{aligned}$$

The result follows by applying the shrinking hypothesis. □

Proof of Lemma 3.4.2 Let $T\subseteq J_S$ and $h\in T$ be such that $\phi^*_{ih}=1$, $d^*_h\leq Q$. Let us denote $S=T\setminus\{h\}$. Because h is linked only to facility i, we have $\omega^*((I\setminus\{l\})\cup\overline{S}:S)=\omega^*((I\setminus\{l\})\cup\overline{T}:T)$. We also have $\phi^*(I\setminus\{l\}:S)=\phi^*(I\setminus\{l\}:T)-1$. It follows that $\omega^*((I\setminus\{l\})\cup\overline{S}:S)+2\phi^*(I\setminus\{l\}:S)=\omega^*((I\setminus\{l\})\cup\overline{T}:T)+2\phi^*(I\setminus\{l\}:T)-2$. If T and k violate a BFCI then $\omega^*((I\setminus\{l\})\cup\overline{T}:T)+2\phi^*(I\setminus\{l\}:T)<2r(T,\{l\})\leq 2(r(S,\{l\})+1)$ and the result follows. □

Proof of Lemma 3.4.4 S_1,S_2 are not connected between them nor with facility i, i.e., $x^*(S_1:S_2)=x^*(i:S_2)=y^*(i:S_2)=0$. Suppose that (i,S) defines a violated BFCI, i.e.,

$$x^*((I\setminus\{i\}:\overline{S}:S)+2y^*(I\setminus\{i\}:S)<2r(S,\{i\}).$$

But given that S_1 and S_2 lie in different connected components we have

$$x^*((I \setminus \{i\} : \overline{S} : S) + 2y^*(I \setminus \{i\} : S) = x^*(\delta(S_2)) + 2y^*(I : S_2) \\ + x^*((I \setminus \{i\}) \cup \overline{S}_1 : S_1) + 2y^*(I \setminus \{i\} : S_1).$$

Joining both relationships and taking into account that S_2 satisfies the CC we have

$$\begin{aligned} x^*((I \setminus \{i\}) \cup \overline{S}_1 : S_1) + 2y^*(I \setminus \{i\} : S_1) &< 2r(S, \{i\}) - [x^*(\delta(S_2)) + 2y^*(I : S_2)] \\ &\leq 2r(S, \{i\}) - 2r(S_2) \\ &\leq 2r(S_1, \{i\}) \end{aligned}$$

and the result follows. □

Proof of Proposition 3.4.5 Let $S \subseteq J_S$ be a customer set in the shrunk graph *crossing* the set $\{u,v\}$, i.e., $S \cap \{u,v\}, S \setminus \{u,v\}, \{u,v\} \setminus S \neq \emptyset$. Without loss of generality, we suppose that $u \in S, v \notin S$. We will show that the set $T = S \cup \{v\}$ induces a violation of value at least the same as that induced by S. First note that if u or v take the role of nodes h or j in inequality (3.6) then it will not be violated. As a consequence of this, nodes that can take the place of h or j are among those that have not been shrunk. Let us compute the left-hand side of inequality (3.6) for S and T, that we denote as $\alpha(S)$ and $\alpha(T)$, respectively, and see that they satisfy the following relationship:

$$\begin{aligned} \alpha(T) &= \alpha(S) + \omega^*(\delta(v)) - 2\omega^*(v : S) \\ &\leq \alpha(S). \end{aligned}$$

As the right hand side of the inequality is the same for both S and T, the violation incurred by set T is bigger than that of S and the result follows. □

CHAPTER 4

A BRANCH-AND-CUT-AND-PRICE ALGORITHM

Notes about the chapter

The contents of this chapter correspond to those of the article entitled *A Branch-And-Cut-And-Price Algorithm for the Capacitated Location-Routing Problem*, co-authored with Professors Jean-François Cordeau and Bernard Gendron, which has been submitted for publication to INFORMS Journal on Computing (ISSN: 1526-5528), in July 2011. Preliminary results have also been presented in the *TRANSLOG, Transportation and Logistics Conference*, in Viña del Mar, Chile (2009) and in the *XXIV EURO Conference*, in Lisbon, Portugal (2010).

A Branch-And-Cut-And-Price Algorithm for the Capacitated Location-Routing Problem

Claudio Contardo[1,3], Jean-François Cordeau[2,3], Bernard Gendron[1,3]

[1]Département d'informatique et de recherche opérationnelle, Université de Montréal
C.P. 6128, succ. Centre-ville, Montréal (PQ) Canada H3C 3J7

[2]Canada Research Chair in Logistics and Transportation and HEC Montréal
3000 chemin de la Côte-Sainte-Catherine, Montréal (PQ) Canada H3T 2A7

[3]Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT)
C.P. 6128, succ. Centre-ville, Montréal (PQ) Canada H3C 3J7

In this paper we present an exact algorithm for the Capacitated Location-Routing Problem (CLRP) based on column and cut generation. The CLRP is formulated as a set-partitioning problem which also inherits all of the known valid inequalities for the flow formulations of the CLRP. We introduce five new families of inequalities that are shown to dominate some of the cuts from the two-index formulation. The problem is then solved by column generation, where the sub-problem consists in finding a shortest path of minimum reduced cost under capacity constraints. We first use the two-index formulation for enumerating all of the possible subsets of depot locations that could lead to an optimal solution of cost smaller than or equal to a given upper bound. For each of these subsets, the corresponding Multiple Depot Vehicle Routing Problem is solved by means of column generation. The results show that we can improve the bounds found in the literature, solve to optimality some previously open instances, and improve the upper bounds on some other.

Key words: location-routing, vehicle routing, branch-and-cut-and-price, column generation.

4.1 Introduction

In the Capacitated Location-Routing Problem (CLRP) we are given a set I of potential facilities and a set J of customers. With every facility $i \in I$ are associated a fixed opening cost f_i and a capacity b_i. To every customer $j \in J$ is associated a demand d_j. Distances are assumed to be symmetric. The problem can thus be defined on an undirected graph $G = (V, E)$, where $V = I \cup J$ is the vertex set and E is the edge set. To every edge $e = \{i, j\}$ we associate a routing cost c_{ij}. The fleet is assumed to be of unlimited size and homogeneous, each vehicle having a capacity Q. The objective is to choose a subset of facilities and to construct vehicle routes around these facilities to visit every customer exactly once, respecting both vehicle and facility capacities while minimizing the sum of fixed costs and routing costs.

The CLRP arises in several real-world applications. Labbé and Laporte [86] solve the problem of locating postal boxes while minimizing a linear combination of routing costs (those of the mail collecting trucks) and customer inconvenience produced by their distance to the nearest postal box. Billionet et al. [21] consider a location problem arising in mobile networks. The problem consists in locating radio-communication stations, designing rings and building antennaes inside these rings at minimum cost. Gunnarsson et al. [71] solve a location-routing problem arising in the pulp distribution industry in Scandinavia.

The CLRP can be formulated as a three-index mixed-integer program [115]. In such a formulation, asymmetries in the distance matrix and heterogeneities in the vehicle capacities can be easily taken into account. However, due to the large number of variables and its poor linear programming relaxation it has no practical use within an enumeration method such as branch-and-bound. In the context of exact algorithms for solving the CLRP, Belenguer et al. [19] developed a two-index formulation and proposed several families of valid inequalities, such as y-Capacity Cuts (y-CC), Path Constraints (PC), Facility Degree Constraints (FDC), Imparity Constraints (IC) and Facility Capacity Inequalities (FCI). They solve the problem by means of branch-and-cut and their algorithm succeeds in solving small and medium size instances with up to 50 customers. Con-

tardo et al. [36] introduced three new formulations of the CLRP based on vehicle flows and commodity flows. They introduced strengthenings of the FCI as well as Location-Routing Comb Inequalities (LR-CI), Location-Routing Generalized Large Multistar Inequalities (LRGLM) and y-Generalized Large Multistar Inequalities (y-GLM), exploiting the fact that facilities have limited capacities. Their algorithms were able to solve instances containing up to 100 customers, the largest for branch-and-cut methods. Akca et al. [4] developed a set-partitioning formulation based on a Dantzig-Wolfe decomposition of the three-index model. They solve the problem by means of branch-and-price, where the subproblem is a shortest path problem under capacity constraints (SPPRC). Their formulation provides reasonably good bounds at the root node of the search tree but does not appear to be effective for closing the gap using branching. Baldacci et al. [16] also formulate the CLRP as a set-partitioning problem. They use three different relaxations of the formulation that are applied sequentially in an additive manner. In the last step, they solve a small number of MDVRP by means of a cut-and-price-and-branch method, in which the root node is solved by colum generation, and then enumerate all of the remaining columns whose reduced cost is smaller than a given gap. The resulting integer program is then solved by means of a general-purpose integer programming solver. They use a stregthened version of the CC as well as clique inequalities. The bounds provided by their model are very tight, being able to solve instances with up to 199 customers and 14 facilities.

The CLRP is $\mathscr{NP}$-hard as it generalizes both the Capacitated VRP (CVRP) and the Capacitated Facility Location Problem (CFLP). Moreover, the presence of capacities for both the vehicles and the facilities makes it particularly hard. Because of this, solution approaches for solving medium and large size instances have mainly focused on the development of heuristics. These heuristics in most cases use some decomposition scheme to divide the problem into a design sub-problem for the location decisions and an operational sub-problem for the routing part [77, 97, 99, 115, 146]. Recently, Prins et al. [120, 121, 122] have proposed several metaheuristics that include memetic algorithms, cooperative Lagrangean relaxation with tabu search and greedy randomized adaptive search procedure (GRASP). Computational experience shows that the second

approach is the most effective one for tackling large instances of the CLRP.

The contributions of this paper can be summarized as follows:

i. We adapt the set-partitioning formulation due to Akca et al. [4] so that all of the cuts valid for the two-index formulation of the CLRP [19, 36] can be easily incorporated.

ii. We introduce two bounding procedures that are applied sequentially and that allow, in most cases, to reduce the CLRP to a series of multiple depot VRP, as in Baldacci et al. [16]. Our computational results show that our bounding procedures can be stronger than those of Baldacci et al. [16] for some instances.

iii. We introduce several new families of cuts that are effective for closing the optimality gap. Moreover, our computational experience shows that using state-space relaxation in the pricing problem suffices to get bounds close to those obtained by pricing on elementary routes (routes that do not contain cycles).

iv. We introduce a new fathoming rule that accelerates the solution of the pricing subproblems.

As a result, our algorithm is able to solve all instances that are also solved by the exact method of Baldacci et al. [16] as well as four previously open instances. Additionally, we improve the best known feasible solution for three other instances. Moreover, for the instances that remain unsolved we improve the best known lower bounds.

The paper is organized as follows. In Section 4.2 we present some formulations of the CLRP, namely the two-index vehicle-flow formulation due to Belenguer et al. [19] as well as the set-partitiong formulation due to Akca et al. [4]. In Section 4.3 we describe the valid inequalities used through this paper. It includes some known valid inequalities from the two-index formulation and the set-partitioning problem as well as new valid inequalities that are shown to be valid for the set-partitioning formulation of the CLRP. In Section 4.4 we describe the exact algorithm used to solve the CLRP to optimality. We first describe the separation algorithms used to find violated valid inequalities. We then describe the different bounding procedures as well as the pricing algorithms used to solve the corresponding set-partitioning problems. Finally, we discuss some computational

issues that are mostly implementation-specific and that have an important impact on the performance of the algorithm. In Section 4.5 we present our computational results and compare against the state-of-the-art solvers for solving the CLRP. We conclude in Section 4.6 with a summary of the proposed methodology and discuss possible avenues of future research.

4.2 CLRP Formulations

In this section we first present the two-index vehicle-flow formulation of the CLRP due to Belenguer et al. [19] and the set-partitioning formulation introduced by Akca et al. [4]. We also show that any inequality valid for the two-index formulation can be easily extended to the set-partitioning formulation.

4.2.1 Two-index vehicle-flow formulation

Belenguer et al. [19] proposed the following two-index vehicle-flow formulation for the CLRP. For every vertex set U, let $\delta(U)$ be the edge subset containing all those edges with exactly one endpoint in U. For two disjoint vertex sets T,U, let $(T:U)$ be the edge subset containing all edges with one endpoint in T and the other in U. For every facility $i \in I$, let z_i be a binary variable equal to 1 iff facility i is selected for opening. For every edge $e \in E$, let x_e be a binary variable equal to 1 iff edge e is traversed once by some vehicle. Finally, for every edge $e \in \delta(I)$ let y_e be a binary variable equal to 1 iff edge e is used twice by some vehicle. For a given edge set $F \subseteq E$ let $x(F) = \sum_{e \in F} x_e$, $y(F) = \sum_{e \in F} y_e$. For a given customer subset $S \subseteq J$, let $d(S) = \sum_{j \in S} d_j$ and $r(S) = \lceil d(S)/Q \rceil$ (which actually is a lower bound on the number of vehicles needed to serve the customers in S). The formulation is the following.

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{e \in E} c_e x_e + 2 \sum_{e \in \delta(I)} c_e y_e \qquad \text{(TIF)}$$

subject to

$$x(\delta(j)) + 2y(I : \{j\}) = 2 \qquad j \in J \qquad (4.1)$$

$$x(\delta(S)) + 2y(I : S) \geq 2r(S) \qquad S \subseteq J, |S| \geq 2 \qquad (4.2)$$

$$x_{ij} + y_{ij} \leq z_i \qquad i \in I, j \in J \qquad (4.3)$$

$$x(I : \{j\}) + y(I : \{j\}) \leq 1 \qquad j \in J \qquad (4.4)$$

$$x((I \setminus \{i\}) \cup \overline{S} : S) + 2y(I \setminus \{i\} : S) \geq 2 \qquad i \in I, S \subseteq J, d(S) > b_i \qquad (4.5)$$

$$x(\delta(S)) \geq 2(x(\{h\} : I') + x(\{j\} : I \setminus I')) \qquad S \subseteq J, |S| \geq 2, h, j \in S, I' \subset I \qquad (4.6)$$

$$z_i \in [0,1] \text{ and integer} \qquad i \in I \qquad (4.7)$$

$$x_e \geq 0 \text{ and integer} \qquad e \in E \qquad (4.8)$$

$$y_e \geq 0 \text{ and integer} \qquad e \in \delta(I). \qquad (4.9)$$

Constraints (4.1) are the degree constraints at customer nodes. Constraints (4.2) are capacity cuts (CC), whose role is to forbid at the same time proper tours disconnected from facilities and tours serving a demand larger than Q. Constraints (4.3) ensure that there is no outgoing flow leaving from closed facilities. Constraints (4.4) are the path constraints for single customers. They forbid routes of the form $i_1 \rightarrow j \rightarrow i_2, i_1, i_2 \in I, i_1 \neq i_2, j \in J$. Constraints (4.5) are the facility capacity inequalities (FCI). They forbid the existence of routes leaving from a same facility i and serving a demand larger than b_i. Constraints (4.6) are the path constraints (PC) for multiple customers. Their role is to prevent the route of a single vehicle from joining two different facilities.

Belenguer et al. [19] have shown that constraints (4.2) can be strengthened into the so-called y-Capacity Cuts (y-CC):

$$x(\delta(S)) + 2y(I : S \setminus S') \geq 2r(S) \qquad S \subseteq J, |S| \geq 2, S' \subset S, r(S \setminus S') = r(S). \qquad (4.10)$$

These authors showed that the FCI can be generalized to take into account several facilities in the same constraint. For a subset $I' \subseteq I$ of facilities, they define $r(S, I') = \lceil (d(S) - b(I'))/Q \rceil$ (which is a lower bound on the number of vehicles that are needed

to serve the demand of customers in S from facilities other than those in I'), where $b(I') = \sum_{i \in I'} b_i$. The following constraint, introduced by Contardo et al. [36] and called strengthened FCI (SFCI), takes into account this observation and can be shown to dominate the FCI as well as the SFCI introduced by Belenguer et al. [19]:

$$x(I \setminus I' : S) + 2y(I \setminus I' : S \setminus S') \geq 2r(S, I') \qquad S \subseteq J, I' \subseteq I, S \subset S', r(S \setminus S', I') = r(S, I'). \tag{4.11}$$

4.2.2 Set-partitioning formulation

We now describe the set-partitioning formulation introduced by Akca et al. [4] and also used by Baldacci et al. [16], and link it to the two-index vehicle-flow formulation so that all of the known cuts for the CLRP are also valid. Let us denote by Ω_i the set of all routes (possibly containing cycles) starting and ending at facility $i \in I$ and servicing a subset of at least two customers with total demand of Q or less, and let $\Omega = \cup_{i \in I} \Omega_i$ be the set of all possible routes servicing two or more customers with total accumulated demand of Q or less. For every $l \in \Omega$ let us associate a binary variable λ_l equal to 1 if l appears in the optimal solution of the CLRP and 0 otherwise, and a cost c_l for using this route. For every edge $e \in E$ and route $l \in \Omega$ let q_l^e be the number of times that edge e appears in route l. If Ω is restricted to contain only elementary routes then q_l^e is a binary constant, otherwise it can be a general integer. On the other hand, let us define binary variables y_{ij} for $\{i, j\} \in \delta(I)$ equal to 1 iff customer j is served from facility i by a single-customer route. Note that if distances satisfy the triangular inequality, the optimal solution of this problem will only contain elementary paths even if Ω is enlarged to contain routes with cycles. In fact, in this case it is always possible to build from a solution with cycles, another solution with elementary routes at lower cost. Let us extend the demands to facility nodes by letting $d_v = 0$ for every $v \in I$. A valid formulation for the CLRP is

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{l \in \Omega} c_l \lambda_l + 2 \sum_{e \in \delta(I)} c_e y_e \tag{SPF}$$

subject to

$$\sum_{l \in \Omega} \sum_{e \in \delta(\{j\})} q_l^e \lambda_l + 2y(I : \{j\}) = 2 \qquad j \in J \tag{4.12}$$

$$\sum_{l \in \Omega_i} \sum_{\{h,j\} \in E} (d_h + d_j) q_l^{\{h,j\}} \lambda_l + 2 \sum_{j \in J} d_j y_{ij} \leq 2 b_i z_i \qquad i \in I \tag{4.13}$$

$$\lambda_l \geq 0 \text{ and integer} \qquad l \in \Omega \tag{4.14}$$

$$y_e \geq 0 \text{ and integer} \qquad e \in \delta(I) \tag{4.15}$$

$$z_i \in [0,1] \text{ and integer} \qquad i \in I \tag{4.16}$$

In this formulation, constraints (4.12) ensure that each customer is served exactly once. Constraints (4.13) are the facility capacity inequalities. They ensure that the demand served from any facility i will not exceed its capacity b_i. The distinction between single-customer and multiple-customer routes naturally defines a relationship between vehicle-flow variables x from the two-index formulation and λ, as follows

$$\sum_{l \in \Omega} q_l^e \lambda_l - x_e = 0 \qquad e \in E \tag{4.17}$$

In such a way, all of the valid inequalities from the two-index formulation of the CLRP can be translated into the set-partitioning formulation by using identities (4.17).

4.3 Valid inequalities

In this section we describe the valid inequalities that can be applied to formulation (SPF) and that strengthen the LP relaxation. First, we describe some of the valid inequalities that have been developed in the context of the two-index and three-index formulations by Belenguer et al. [19] and Contardo et al. [36]. We then describe new

families of valid inequalities that are shown to dominate several of the former and that effectively strengthen formulation (SPF).

4.3.1 Valid inequalities for the two-index formulation

The valid inequalities for formulation (TIF) include several different families. After a series of preliminary tests, we have decided to keep only a subset of them, namely strengthened comb inequalities (SCI), framed capacity inequalities (FrCI), effective strengthened facility capacity inequalities (ESFCI), facility degree constraints (FDC) and location-routing comb inequalities (LR-CI). To include any of these constraints into formulation (SPF) we use identity (4.17). For details on the inequalities we refer to Lysgaard et al. [101], Belenguer et al. [19] and Contardo et al. [36].

4.3.2 Valid inequalities for the set-partitioning formulation

The valid inequalities for the set-partitioning formulation include a strengthening of the y-SCC introduced by Baldacci et al. [14] for solving the CVRP and also used by Baldacci et al. [16] for the CLRP. We also introduce strengthenings of the degree constraints (4.12), of SFCI constraints (4.11), ESFCI and FrCI. We complement this with the addition of subset-row inequalities (SRI).

Any constraint in the two-index space can be translated into a constraint in the route space by using identity (4.17). However, the constraints translated this way will not take into account the fact that a route can cross more than once a given subset of vertices. For a given subset of routes $\mathscr{R} \subseteq \Omega$, let us define $\lambda(\mathscr{R}) = \sum_{l \in \mathscr{R}} \lambda_l$. We also let $i(l)$, $J(l)$ and $E(l)$ to be the facility to which l is assigned, the set of customers served by l and the set of edges used by l, respectively.

4.3.2.1 y-Strengthened CC (y-SCC)

Let us consider, for a given customer set $S \subseteq J$ and subset $S' \subset S$ such that $r(S \setminus S') = r(S)$, the corresponding y-CC as described in Belenguer et al. [19] and Contardo et al. [36], for formulations (TIF) and (SPF), respectively:

$$x(\delta(S)) + 2y(I : S \setminus S') \geq 2r(S) \qquad \text{for TIF} \tag{4.18}$$

$$\sum_{l \in \Omega} \sum_{e \in \delta(S)} q_l^e \lambda_l + 2 \sum_{e \in [I:S \setminus S']} y_e \geq 2r(S) \qquad \text{for SPF.} \tag{4.19}$$

Baldacci et al. [14] noted that the CC (4.2) can be strengthened by setting the coefficient of a given path variable λ_l to be 0 if l does not serve a customer in S and 1 otherwise, rather than counting the number of edges of l that are also in $\delta(S)$. For formulation (SPF), the constraint is the following,

$$\lambda(\{l : J(l) \cap S \neq \emptyset\}) + y(I : S) \geq r(S). \tag{4.20}$$

For the y-CC we can apply the same reasoning, as stated in the following proposition.

Proposition 4.3.1. *Let $S \subseteq J$ be a subset of customers, and $S' \subset S$ such that $r(S \setminus S') = r(S)$, the following constraint is valid for the CLRP and dominates the y-CC* (4.19) *and the SCC* (4.20)*.*

$$\lambda(\{l : J(l) \cap S \neq \emptyset\}) + y(I : S \setminus S') \geq r(S). \tag{4.21}$$

We call this constraint the y-Strengthened CC (y-SCC).

Proof Let us define $\mathscr{L}(T) = \{l \in \Omega : J(l) \cap T \neq \emptyset\}$, for $T \subseteq J$. We then have

$$\lambda(\mathscr{L}(S)) = \lambda(\mathscr{L}(S \setminus S')) + \lambda(\mathscr{L}(S')) - \lambda(\mathscr{L}(S \setminus S') \cap \mathscr{L}(S')). \tag{4.22}$$

We have to prove that

$$\lambda(\mathscr{L}(S)) + y(I : S \setminus S') \geq r(S)$$

is a valid inequality of the CLRP. In fact, we have

$$\begin{aligned}\lambda(\mathscr{L}(S))+y(I:S\setminus S') &= \lambda(\mathscr{L}(S\setminus S'))+\lambda(\mathscr{L}(S'))-\lambda(\mathscr{L}(S\setminus S')\cap\mathscr{L}(S'))+y(I:S\setminus S')\\ &\geq r(S\setminus S')+\lambda(\mathscr{L}(S'))-\lambda(\mathscr{L}(S\setminus S')\cap\mathscr{L}(S'))\\ &\geq r(S\setminus S')\\ &= r(S).\end{aligned}$$

The dominance with respect to the y-CC comes from the fact that a route l that visits a customer set S must have two or more edges crossing it, and the dominance with respect to the SCC comes from the consideration of the customer set S'. □

4.3.2.2 Strengthened Degree Constraints (SDEG)

Degree constraints in the two-index space count the number of times that a certain node is traversed. If a node can be traversed several times by a single route, then a stronger version of the degree constraint is

$$\lambda(\{l\in\Omega : j\in J(l)\})+y(I:\{j\})\geq 1 \qquad\qquad j\in J. \tag{4.23}$$

These constraints are relevant when, instead of restricting the state-space to elementary routes, it is rather relaxed to contain routes with cycles. In our algorithm we have found that the addition of these constraints when pricing on non-elementary routes is an effective method to get bounds close to the ones obtained by pricing on elementary routes. Indeed, the problem of finding an appropriate balance between speed and lower bound quality for different variants of the SPPRC has already been studied and is a key aspect in the performance of column generation based algorithms for vehicle routing problems [see, e.g., 23, 49, 128]. This intuition is supported by the following proposition,

Proposition 4.3.2. *The optimal value of the linear relaxation of* (SPF) *when restricting the space* Ω *to elementary routes is the same as when* Ω *is enlarged to routes with cycles after adding the SDEG constraints* (4.23).

Proof Obviously elementary routes satisfy constraints (4.23), so the value of the linear relaxation on the elementary case is at least as good as in the relaxed case. On the other hand, in the relaxed case, no route with cycles will be basic after the addition of (4.23). Indeed, let $j \in J$ be any customer, and let $\Omega_j, \Omega_{cyc(j)}$ be the subsets of routes traversing j and containing a cycle in j, respectively (obviously $\Omega_{cyc(j)} \subseteq \Omega_j$). For that customer, from constraints (4.12) we have

$$\sum_{l \in \Omega_j \setminus \Omega_{cyc(j)}} \sum_{e \in \delta(\{j\})} q_l^e \lambda_l = 2 - 2y(I : \{j\}) - \sum_{l \in \Omega_{cyc(j)}} \sum_{e \in \delta(\{j\})} q_l^e \lambda_l \tag{4.24}$$

Using (4.23), $\Omega_{cyc(j)}$ also has to satisfy

$$\sum_{l \in \Omega_j \setminus \Omega_{cyc(j)}} \lambda_l \geq 1 - y(I : \{j\}) - \sum_{l \in \Omega_{cyc(j)}} \lambda_l \tag{4.25}$$

After multiplying the second equation by two, the left-hand side of both equations coincide, and the following relationship holds between their right-hand sides

$$\sum_{l \in \Omega_{cyc(j)}} \sum_{e \in \delta(\{j\})} q_l^e \lambda_l \leq \sum_{l \in \Omega_{cyc(j)}} 2\lambda_l \tag{4.26}$$

As $\sum_{e \in \delta(\{j\})} q_l^e \geq 4$ for $l \in \Omega_{cyc(j)}$ (because j is traversed at least twice, i.e. by at least 4 edges), it follows that $\lambda_l = 0$ for every $l \in \Omega_{cyc(j)}$. □

4.3.2.3 Set-Partitioning SFCI (SP-SFCI)

Let us consider the SFCI constraints (4.11), and let S, I' and S' be as in (4.11). The following strengthening of the SFCI, called Set-Partitioning SFCI (SP-SFCI), is valid for the CLRP and dominates (4.11):

$$\sum_{k \in I \setminus I'} \lambda(\{l \in \Omega_k : J(l) \cap S \neq \emptyset\}) + y(I \setminus I' : S \setminus S') \geq r(S, I'). \tag{4.27}$$

Before proving the validity of the above constraint, let us define some notation. For each $i \in I, j \in S$, let w_{ij} be a binary constant equal to 1 iff customer j is served from

facility i. Let $W_{I'} = \{j \in J : w_{ij} = 1 \text{ for some } i \in I'\}$. For given subsets $H \subseteq I$ and $S \subseteq J$, let us define $\mathscr{L}_H(S) = \cup_{i \in H}\{l \in \Omega_i : J(l) \cap S \neq \emptyset\}$. Now, let us prove the validity of constraints (4.27).

Proposition 4.3.3. *Constraints* (4.27) *are valid for the CLRP and dominate the SFCI* (4.11).

Proof Let us consider first the case $S' = \emptyset$. Indeed, if $S \subseteq W_{I'}$ then constraint (4.27) is trivially satisfied (because $r(S,I') = 0$). If $S \subseteq \overline{W}_{I'}$ then $\lambda(\mathscr{L}_{I'}(S)) = y(I' : S) = 0$ and therefore $\lambda(\mathscr{L}_{I\setminus I'}(S)) + y(I \setminus I' : S) = \lambda(\mathscr{L}_I(S)) + y(I : S) \geq r(S) \geq r(S,I')$. If $S \cap W_{I'} \neq \emptyset$ and $S \cap \overline{W}_{I'} \neq \emptyset$, we have $\lambda(\mathscr{L}_{I\setminus I'}(S)) + y(I \setminus I' : S) = \lambda(\mathscr{L}_{I\setminus I'}(S \cap W_{I'})) + y(I \setminus I' : S \cap W_{I'}) + \lambda(\mathscr{L}_{I\setminus I'}(S \cap \overline{W}_{I'})) + y(I \setminus I' : S \cap \overline{W}_{I'}) \geq r(S \cap W_{I'}, I') + r(S \cap \overline{W}_{I'}) \geq r(S,I')$. Let us suppose now that $S' \neq \emptyset$. Let $S'' \subseteq S'$ be such that $y(I \setminus I' : S') = |S''|$, i.e., the customers in S' that are served by single-customer routes from facilities in $I \setminus I'$ are exactly those in S''. As a consequence of this, $\lambda(\mathscr{L}_{I\setminus I'}(S)) = \lambda(\mathscr{L}_{I\setminus I'}(S \setminus S''))$ and then $\lambda(\mathscr{L}_{I\setminus I'}(S)) + y(I \setminus I' : S \setminus S') = \lambda(\mathscr{L}_{I\setminus I'}(S \setminus S'')) + y(I \setminus I' : S \setminus S'') \geq r(S \setminus S'', I') = r(S,I')$. The dominance with respect to constraints (4.11) comes from the fact that *i*) routes crossing set S several times are only counted once, and *ii*) edges connecting S with $\overline{S}$ are considered only if they belong to routes departing from $I \setminus I'$. □

4.3.2.4 Set-Partitioning ESFCI (SP-ESFCI)

The Effective SFCI were introduced by Belenguer et al. [19] and Contardo et al. [36] and are valid for the two-index formulation (TIF). They can be seen as a strengthening of the SFCI by noticing that the right-hand side of such constraint can be in fact lifted whenever $z_i = 0$ for some $i \in I'$. For the set-partitioning formulation (SPF) they can be written as

$$\sum_{k \in I \setminus I'} \lambda(\{l \in \Omega_k : J(l) \cap S \neq \emptyset\}) + y(I \setminus I' : S \setminus S') \geq r(S,I') + z_i(r(S,I' \setminus \{i\}) - r(S,I')). \tag{4.28}$$

The validity proof follows from the validity of the SP-SFCI for the two cases $z_i = 1$ and $z_i = 0$.

4.3.2.5 Strengthened Framed Capacity Inequalities (SFrCI)

The framed capacity inequalities were developed by Augerat [8] for the CVRP and later succesfully used by other authors in the development of algorithms based on cutting planes and column generation [59, 101]. Given a customer set S, that we call the frame, and a partition of it $(S_i)_{i=1}^t$, the related FrCI seen in formulation (TIF) is

$$x(\delta(S)) + 2y(I:S) + \sum_{i=1}^{t}(x(\delta(S_i)) + 2y(I:S_i)) \geq 2\left(BPP(S|(S_i)_{i=1}^t) + \sum_{i=1}^{t} r(S_i)\right), \tag{4.29}$$

where $BPP(S|(S_i)_{i=1}^t)$ represents the solution of the following bin-packing problem. For every $i = 1,\ldots,t$ consider $\lceil d(S_i)/Q \rceil$ items of size Q except for the last item that will have size $d(S_i) - (\lceil d(S_i)/Q \rceil - 1)Q$. Also, set the bins to have size Q. In addition to using identity (4.17) to adapt this constraint to formulation (SPF), the same observation as done for the y-SCC, SDEG, SP-SFCI and SP-ESFCI can be applied. The following constraint, called strengthened FrCI (SFrCI) is valid for the CLRP and also dominates the FrCI.

$$\lambda(\{l \in \Omega : J(l) \cap S \neq \emptyset\}) + \sum_{i=1}^{t} \lambda(\{l \in \Omega : J(l) \cap S_i \neq \emptyset\}) + 2y(I:S) \geq BPP(S|(S_i)_{i=1}^t) + \sum_{i=1}^{t} r(S_i). \tag{4.30}$$

Before proving the validity of the SFrCI we need the following lemma

Lemma 4.3.4 (Augerat [8])**.** *Let $S \subseteq J$ and $(S_i)_{i=1}^t$ a partition of S. If $\lceil d(S_1 \cup S_2)/Q \rceil = \lceil d(S_1)/Q \rceil + \lceil d(S_2)/Q \rceil$ then $BPP(S|S_1, S_2, \ldots, S_t) \geq BPP(S|S_1 \cup S_2, S_3, \ldots, S_t)$. Otherwise $BPP(S|S_1, S_2, \ldots, S_t) + 1 \geq BPP(S|S_1 \cup S_2, S_3, \ldots, S_t)$.*

Proof See Augerat [8]. □

Proposition 4.3.5. *Constraints* (4.30) *are valid for* (SPF)*.*

Proof The proof uses exactly the same arguments as in Augerat [8]. Let us suppose first that sets S_i satisfy $d(S_i) \leq Q$. Let us consider the bin-packing problem defined above, with objects of sizes $d(S_i)$ for every $i = 1,\ldots,t$ and bin size equal to Q. Let us denote the set of objects by K. In this context, let us call a *cut of object k in K* the following operation: remove k (of size $d(k)$) from K and replace it by two smaller objects whose total size is equal to $d(k)$. It is known that after a cut operation, the solution of a BPP is reduced by at most one unit. As a consequence, the same applies for q cut operations, so that the solution of the BPP is reduced by at most q units. In the case of the CLRP, the quantity $w = \sum_{i=1}^{t}(\lambda(\{l \in \Omega : J(l) \cap S_i \neq \emptyset\}) + y(I : S_i) - 1)$ represents exactly the number of cuts that are applied to the set S, and thus $BPP(S|(S_i)_{i=1}^{t}) + w$ represents a lower bound on the number of vehicles needed to serve the demand of S. Now, in the general case, let (λ, y, z) be a solution of (SPF). For every subset S_i, (λ, y) define a partition $S_i^k, k = 1,\ldots n_i$ of subsets of S_i such that *i)* $\lambda(\{l \in \Omega : J(l) \cap S_i^k \neq \emptyset\}) + y(I : S_i^k) = 1$ and *ii)* $n_i = \lambda(\{l \in \Omega : J(l) \cap S_i \neq \emptyset\}) + y(I : S_i)$. From the first case we have that $\lambda(\{l \in \Omega : J(l) \cap S \neq \emptyset\}) + y(I : S) \geq BPP(S|(S_1^k)_{k=1}^{n_1}, (S_2^k)_{k=1}^{n_2}, \ldots, (S_t^k)_{k=1}^{n_t})$. For every $i = 1,\ldots,t$ we apply n_i successive contractions of the subsets S_i^k and compute $\alpha(i,j)$ equal to the number of times that $BPP(S|(S_1^k)_{k=1}^{n_1}, (S_2^k)_{k=1}^{n_2}, \ldots, (S_t^k)_{k=1}^{n_t})$ decreases by one unit after a contraction. By applying the lemma, we have that $\alpha(i,1) = \lceil d(S_i^1)/Q \rceil + \lceil d(S_i^2)/Q \rceil - \lceil d(S_i^1 \cup S_i^2)/Q \rceil = 2 - \lceil d(S_i^1 \cup S_i^2)/Q \rceil$ and, more generally, $\alpha(i,j) = j + 1 - \lceil d(\bigcup_{k=1}^{j} S_i^k)/Q \rceil$. At the end of all of these successive contractions we will have that $\lambda(\{l \in \Omega : J(l) \cap S \neq \emptyset\}) + y(I : S) \geq BPP(S|(S_i)_{i=1}^{t}) - \sum_{i=1}^{t}(n_i - \lceil d(S_i)/Q \rceil)$ □

4.3.2.6 Subset-Row Inequalities (SRI, Jepsen et al. [80])

The subset-row inequalities are a special case of the clique inequalities [11] and are valid for the set partitioning formulation of the CLRP. Let us consider the conflict graph $\mathscr{H}_\lambda$ constructed as follows. The vertices of $\mathscr{H}_\lambda$ are the routes $l \in \Omega$ such that $\lambda_l > 0$. Two vertices in $V(\mathscr{H}_\lambda)$ are linked by an edge if they share at least one customer. A clique in $\mathscr{H}_\lambda$ is a maximal complete induced subgraph of $\mathscr{H}_\lambda$. For every clique $\mathscr{C} \subseteq \mathscr{H}_\lambda$, the

following clique inequality is valid for the CLRP:

$$\sum_{v \in V(\mathscr{C})} \lambda_v \leq 1. \tag{4.31}$$

The addition of clique inequalities into the master problem SPF has, however, an important drawback: they make the pricing problem of finding routes (with or without cycles) of negative reduced cost much more difficult. Indeed, during the pricing problem it must be checked if a partial path participates or not in a clique. This is equivalent to checking if a partial path intersects every column already in a clique in at least one customer node, which in practice is difficult to do. Jepsen et al. [80] introduced the subset-row inequalities. A subset-row inequality is a clique inequality associated to a clique $\mathscr{C}$ to which we assign a subset of customers $\chi(\mathscr{C}) \subseteq J$ such that every column in $\mathscr{C}$ intersects $\chi(\mathscr{C})$ in at least a certain number of customers. If $|\chi(\mathscr{C})|$ is small, the pricing problem can be accelerated as only $|\chi(\mathscr{C})|$ comparisons are needed to check if a given path participates in the clique. These inequalities are a particular case of the clique inequalities and in general provide slightly weaker bounds. The results obtained by Jepsen et al. [80] for the particular case of $|\chi(\mathscr{C})| = 3$ show that the gain for considering the clique inequalities instead of the subset-row inequalities is usually not worth the extra computational effort.

4.4 Solution Methodology

In this section we describe the exact algorithm that solves the CLRP to optimality. We first describe the separation algorithms used in order to find violated inequalities. Then, we describe two bounding procedures that are applied sequentially. The first procedure is based on the two-index formulaton (TIF) with additional cuts. The second procedure is based on the set partitioning formulation (SPF) with additional cuts. We then describe an enumeration procedure to close the optimality gap that is applied only in certain cases. Finally, we describe the computational issues in the implementation of the proposed algorithm.

4.4.1 Separation Algorithms

We now describe the separation algorithms used to separate the different families of valid inequalities used in our algorithm. Our separation strategy is as follows: we first try to generate cuts translated from the two-index formulation (TIF). If no such cuts can be found, we try to generate cuts SDEG, y-SCC, SP-SFCI, SP-ESFCI and SFrCI. If it fails, we try to generate cuts SRI. This strategy allows us to keep the number of strong constraints small as their inclusion in the pricing algorithm make it harder.

4.4.1.1 Inequalities translated from formulation *TIF*

For the valid inequalities translated from the two-index formulation using identity (4.17), such as y-CC, SFCI, ESFCI, SCI, LR-CI or FrCI, we use the separation algorithms introduced by Lysgaard et al. [101], Belenguer et al. [19] and Contardo et al. [36].

4.4.1.2 SDEG, y-SCC, SP-SFCI, SP-ESFCI and SFrCI

Although there is a polynomial number of SDEG constraints, we do not add them all at the beginning of the algorithm, but we rather check if for a certain weak degree constraint, its related strong constraint is violated, and add it to the problem. For the remaining constraints, we use the same principle. In fact, we check if, for any previously found weak constraint y-CC, SFCI, ESFCI or FrCI, its related strong constraint is violated and in this case we add it to formulation SPF.

4.4.1.3 Subset-Row Inequalities

The separation of the subset-row inequalities is done by enumeration just as in Jepsen et al. [80]. Indeed, we only separate SRI for cliques $\mathscr{C}$ such that $|\chi(C)| = 3$. We check for every triplet $(i, j, k) \in J^3$, $i < j < k$ if the corresponding SRI is violated. If it is the case, it is added to the master problem.

4.4.2 First bounding procedure

In this procedure, an enumeration method based on a branch-and-cut algorithm [36] is applied to problem (TIF) after dropping the integrality constraints on the edge variables x and y. This procedure is used to obtain candidate subsets $I' \subseteq I$ of facilities such that the problem restricted to these facilities could lead to a feasible solution with cost smaller or equal than a given upper bound. We denote the set that contains the subsets I' by $\mathscr{I}$. For finding the subsets in $\mathscr{I}$, a good upper bound is needed to prune nodes in the branching tree. In our method, we have used the best feasible solutions found in the literature. For large instances, however, the computation of the whole branching tree can be prohibitive. In this case, the branch-and-bound algorithm is terminated earlier and the uninspected nodes are also added to $\mathscr{I}$. Now, the facilities in a given subset $I' \in \mathscr{I}$ are not only those that are open but also those that could not be fixed in the current node. During the process, different families of valid inequalities are added to strengthen the formulation. However, we only add cuts in nodes whose depth is less than or equal to 5. For each candidate set $I' \subseteq I$ generated by the algorithm we proceed as follows:

i. Based on reduced costs, perform variable fixing on the location variables z, in case set I' contains facilities that remained unfixed.

ii. Based on reduced costs, perform variable fixing on the edge variables x.

iii. Compute the optimal dual variables associated to the degree constraints (4.1).

iv. Compute $K_m(I')$ as an upper bound on the maximum number of routes that serve two or more customers, namely $K_m(I') = \lfloor \max\{\frac{1}{2}x(\delta(I')) : (x,y,z) \in \mathscr{A}\} \rfloor$, where $\mathscr{A}$ stands for the set of constraints (4.1)-(4.9) plus the generated cuts and after dropping the integrality conditions.

For each subset I' found by this algorithm we apply a second bounding procedure and a column enumeration method (in this context, the definition of set I' is implicit and will sometimes be omitted). Note that Baldacci et al. [16] use a similar approach, except that their first bounding procedure computes a global lower bound obtained by solving a relaxation of the set-partitioning problem. This bound is then used to discard non promising subsets $I' \subseteq I$. In Section 4.5 we present computational results comparing

the first bounding procedure that we propose with the one suggested by Baldacci et al. [16].

4.4.3 Second bounding procedure

In this procedure, the following state-space relaxation of formulation (SPF) is solved by means of column generation. Instead of considering elementary routes (i.e., routes without cycles), we allow routes that contain cycles of length three or more, i.e., for nodes $i \neq j \neq k \neq i$ the subpaths $i \rightarrow i$, $i \rightarrow j \rightarrow i$ are forbidden, but the sequence $i \rightarrow j \rightarrow k \rightarrow i$ is permitted. The pricing problem consists in finding routes without cycles of length one or two and such that the reduced costs are minimized. This problem is known in the literature as the 2-cyc-SPPRC [50]. This is an important difference with respect to the method of Baldacci et al. [16] in which the resolution of the subproblem is restricted to elementary routes. During the computation, we add the cuts described in Section 4.3. The violation threshold for the strong cuts is initially set to 0.3. When no more columns of negative reduced cost or violated cuts can be detected, the current objective function value is in fact a valid lower bound for the problem. Let us call this lower bound z^*. We run algorithm ENUM-ESPPRC (described in the next section) in order to price out the remaining columns $l \in \Omega$ such that $\bar{c}_l \leq z_{UB} - z^*$. We have set two hard limits to algorithm ENUM-ESPPRC: the number of labels cannot exceed at any time a maximum $\phi_{max} = 10^6$, and the total number of generated columns cannot exceed $\Delta_{max} = 10^7$. In case of success of this procedure, the columns generated are stored in a column pool $\mathscr{P}$ and the violation threshold for strong constraints is lowered to 0.01. Otherwise, we lower the violation threshold (thus generating more cuts) and continue with the process. This is done at most three times before finishing the column generation process. For instance, for the case of constraints SDEG, the sequence of violation thresholds is $(0.3, 0.25, 0.2, 0.1)$. Whenever the column enumeration ENUM-ESPPRC is done with success, at every following iteration of the column generation method, we do not solve the pricing problem 2-cyc-SPPRC but rather check the reduced costs of columns in $\mathscr{P}$. Note that the size of set $\mathscr{P}$ can be huge and computing the reduced cost of every column in it can be very cumbersome. For dealing with this issue,

at every iteration after the creation of $\mathscr{P}$ in which no columns of negative reduced cost were found, we also delete from the pool all the columns l such that $\overline{c}_l > z_{UB} - z^*$. At the very end of the bounding procedure, we either prune the current node if the final lower bound is greater than or equal to z_{UB}, or otherwise solve the integer problem with the columns generated so far, with the hope of improving the upper bound. In what follows, we first describe the decomposition of the reduced costs for the constraints translated from formulation (TIF), namely all of the constraints in (SPF) plus the cuts that are valid for this formulation. We then show how to incorporate the set-partitioning constraints, such as y-SCC, SDEG, SP-SFCI, SP-ESFCI, SFrCI and SRI into the computation of the reduced costs. We then describe the pricing problem 2-cyc-SPPRC that suits our problem with the additional cuts. We end by describing how we compute lower bounds out of the result of the pricing problem.

4.4.3.1 Decomposition of the reduced costs edge-by-edge

Let us first suppose that only constraints (4.12), (4.13) (with duals α and β, respectively) have been added to the problem. For every $i \in I'$, define the reduced cost of an edge $e \in E(J) \cup \delta(\{i\})$ as

$$\overline{c}_e = \begin{cases} c_e - (\alpha_h + \alpha_j) - (d_h + d_j)\beta_i & \text{if } e = \{h, j\} \in E(J) \\ c_e - \alpha_j - d_j\beta_i & \text{if } e = \{i, j\} \in \delta(\{i\}). \end{cases} \tag{4.32}$$

Let us write a route $l \in \Omega_i$ like a sequence of edges in E, that is $l = (e_t)_{t=1}^{p}$ (in the case in which cycles are permitted, edges may appear more than once in the sequence). Thus, the reduced cost of such a route is given by the following expression:

$$\overline{c}_l = \sum_{t=1}^{p} \overline{c}_{e_t}. \tag{4.33}$$

It follows that in this case a column of minimum reduced cost can be computed as the solution of $|I'|$ shortest path problems with resource constraints. Moreover, the addition

of any cut of the general form

$$\sum_{i\in I'} \tau_i z_i + \sum_{e\in E} \sum_{l\in\Omega} q_l^e \phi_e \lambda_l + \sum_{e\in\delta(I')} \varsigma_e y_e \leq \pi \tag{4.34}$$

produces a contribution to the computation of the reduced cost of the columns that can still be decomposed by edge, thus without breaking the shortest path structure of the pricing. This is the case for all of the cuts valid for the two-index formulation of the CLRP after being translated to formulation (SPF) using identity (4.17).

4.4.3.2 Addition of the strong constraints and effect on the reduced costs

When a constraint cannot be written edge-by-edge, as for constraints (4.21), (4.23), (4.27), (4.28), (4.30) or (4.31), the contribution to the reduced cost cannot be decomposed edge by edge, and thus the original structure of the SPPRC is broken.

Indeed, consider a SRI for a clique C such that for $\chi(C) = \{i, j, k\}$ with dual variable $\sigma \leq 0$. The reduced cost $\bar{c}_l$ of a route $l \in \Omega$ that crosses *at least two* of those three customers must be augmented by $-\sigma$ units.

For the other strong constraints SDEG, y-SCC, SP-SFCI, SP-ESFCI or SFrCI, the contribution to the reduced cost is related to the simple intersection of path l with the sets describing the constraints. For instance, if we consider a SDEG constraint associated to a customer j and with dual variable $\sigma \geq 0$, then the reduced cost of a route l will be reduced by σ units *if l passes through node j*. Now, consider a constraint y-SCC for given sets $S \subseteq J$ and $S' \subset S$ as in (4.21) with dual value $\sigma \geq 0$. The contribution to the reduced cost will reduce it by σ units *if l intersects set S*. For a SP-SFCI or SP-ESFCI associated to sets $S \subset J$, $S' \subset S$, I' with dual variable $\sigma \geq 0$, the contribution to the reduced cost will reduce it by σ units *if route l crosses set S but is not linked to a facility in I'*. Finally, for the SFrCI associated to set S and partition $S_i, i = 1, \ldots, t$ and with dual variable $\sigma \geq 0$, the reduced cost must be reduced by σ units *once for each time that route l intersects either S or any of its subsets*.

4.4.3.3 The pricing problem

The pricing problem corresponds to solve $|I'|$ 2-cyc-SPPRC, one for each facility in I'. The resources associated to each label during the recursion are 1) vehicle load, 2) binary resources related to constraints SDEG, y-SCC, SP-SFCI, SP-ESFCI and SFrCI and 3) resources for taking into account the SRI. The algorithm used to solve these problems is based on dynamic programming (DP), as was done by several authors [14, 50, 56, 80, 128]. Moreover, it is also possible to solve it by means of bidirectional DP (BDP). In classical uni-directional DP, paths are extended until reaching the depot node while ensuring that loads do not exceed capacity. In BDP, however, paths are extended until reaching half of the capacity for later joining paths pairwise. In this section we describe the 2-cyc-SPPRC algorithm used in the context of the CLRP. For general use of the dynamic programming method for solving the SPPRC we refer to the papers cited above. Let us denote by $V(L)$ the set of nodes served by the path represented by label L.

4.4.3.3.1 Resources description As said before, three different types of resources are considered in the problem: vehicle load resource; resources associated to constraints SDEG, y-SCC, SP-SFCI, SP-ESFCI and SFrCI; and resources associated to SRI.

Vehicle load The vehicle load is defined by an integer variable q that keeps track of the load of the current path. It is updated every time that a path is extended to a customer node.

Resources associated to SDEG, y-SCC, SP-SFCI, SP-ESFCI and SFrCI For each of the constraints SDEG, y-SCC, SP-SFCI and SP-ESFCI, the associated resource is defined by a single boolean variable that takes the value *true* if the path intersects the proper set as described before. We designate those sets as critical sets, and denote them by $S(C)$ for every constraint C. For each constraint SFrCI, there will be not one, but as many boolean variables as the size of the partition, plus one for the frame. Each of these variables will take the value *true* if the path crosses the proper set. Now, we do not have one but several critical sets that we denote by $S(C,k)$. Any time that one of these boolean variables passes from *false* to *true*,

the reduced cost of the current path is reduced according to the value of the dual variable.

Resources associated to SRI For every clique C with $\chi(C) = \{i, j, k\}$ we associate three binary variables $r_C(k), k = 1, 2, 3$ that are initialized to 0 until the path crosses one of the customers, in which case the proper variable is set to 1, and the reduced cost of a path will be updated whenever $r_C(1) + r_C(2) + r_C(3)$ reaches the value 2.

4.4.3.3.2 The 2-cyc-SPPRC algorithm We first describe the definition of a label in the recursion of the dynamic programming algorithm. Then, we describe the dominance rules used to discard labels. After that, we describe a fathoming rule that can be aplied in order to also discard labels that cannot lead to a column of negative reduced cost. Next, we describe the path joining procedure to construct feasible paths from a given pair of labels. At the end, we describe the skeleton of the algorithm.

Label definition A label L is defined by

i. A node $v(L)$ which is the end node of the path represented by label L.

ii. A cost $\overline{c}(L)$ representing the reduced cost of the path represented by label L.

iii. A load resource $q(L)$ representing the load of the path represented by label L.

iv. Resources $res_C(L)$ associated to the binding constraints SDEG, y-SCC, SP-SFCI, SP-ESFCI, SFrCI and SRI. For constraints SFrCI and SRI we write $res_C(L,k)$ for the different sub-resources associated to these constraints.

v. An integer variable $v_{dom}(L)$ initially set to -1 and updated whenever L is found to be dominated by a label L', in which case we set $v_{dom}(L) = v(pred(L'))$.

vi. A boolean variable $proc(L)$ initialized to *false* and updated to *true* whenever the algorithm processes the label and inspects its neighbors.

vii. A pointer to the predecessor label $pred(L)$ of L.

viii. A list $succ(L)$ of pointers to the successors of L. $succ_i(L)$ denotes the i-th successor of label L.

Dominance rule Let L, L' be two labels. We denote $\text{SRI}_{LL'} = \{C \in \text{SRI} : \sum_k res_C(L,k) \leq 1$ and $[\sum_k res_C(L',k) \geq 2$ or $\exists\, k$ s.t. $res_C(L,k) < res_C(L',k)]\}$, $n_{C,L,L'} = |\{k : res_C(L,k) < res_C(L',k)\}|$ and $\text{OTH}_{L,L'} = \{C \in \text{SDEG} \cup y\text{-SCC} \cup \text{SP-SFCI} \cup \text{SP-ESFCI} : res_C(L) < res_C(L')\}$. We will say that L is dominated by L' if

i. $v(L) = v(L')$.

ii. $q(L) \geq q(L')$.

iii. $\bar{c}(L) \geq \bar{c}(L') - \sum_{C \in \text{SRI}_{LL'}} \sigma_C + \sum_{C \in \text{SFrCI}} n_{C,L,L'} \sigma_C + \sum_{C \in \text{OTH}_{LL'}} \sigma_C$.

The dominance rule is a direct application of the one used by Archetti et al. [7] for the inclusion of SRI and k-path inequalities in the context of the VRP with split deliveries and time windows (VRPSDTW). A label L that is dominated by another label L' cannot be directly eliminated unless $v(pred(L)) = v(pred(L'))$ or if $v_{dom}(L) \notin \{-1, v(pred(L'))\}$. In that case, label L is removed and recursively we also remove all of its successors in $succ(L)$. Otherwise, $v_{dom}(L)$ is set to $v(pred(L'))$. Note that the inclusion of SDEG constraints allows to weaken the dominance rule with respect to a traditional elementarity constraint, in which the condition for dominance would be $res_C(L) \geq res_C(L')$ for each $C \in \text{SDEG}$.

Fathoming rule In addition to the dominance criterion, a fathoming rule can be applied if a lower bound on the cost of extending a path can be computed. Formally, let L be a label and let $LB(L)$ be a lower bound on the reduced cost that can be obtained by extending L, computed as follows. First of all, discard SRI as their dual variables are negative. For every binding strong constraint $C \in \mathscr{C} =$ SDEG$\cup y$-SCC$\cup$SP-SFCI$\cup$SP-ESFCI$\cup$SFrCI, with dual variables $(\sigma_C)_{C \in \mathscr{C}}$, and for every edge e crossing the critical sets related to these constraints, we decrease the reduced cost of that edge by $\sigma_C/2$ units. We refer to this procedure as *underestimation of constraint C*. As a route that crosses a customer set S must have at least two edges in $\delta(S)$ then the reduced cost of a path computed in this way will in fact be a lower bound on the real reduced cost. We then solve the related 2-cyc-SPPRC with no resources associated to strong constraints, and compute functions f, g and π as follows:

$$f(p,i) = \min\{\overline{c}(L) : v(L) = i, q(L) \leq Q - p + d_i\} \tag{4.35}$$

$$\pi(p,i) = v(pred(arg\,min\{f(p,i)\})) \tag{4.36}$$

$$g(p,i) = \min\{\overline{c}(L) : v(L) = i, q(L) \leq Q - p + d_i, v(pred(L)) \neq \pi(p,i)\} \tag{4.37}$$

For a constraint $C \in$ SFrCI and a customer $i \in J$, let $n_{C,i} = |\{k : i \in S(C,k)\}|$. Also, let

$$h(L) = \begin{cases} f(q(L), v(L)) & \text{if } \pi(q(L), v(L)) \neq v(pred(L)) \\ g(q(L), v(L)) & \text{otherwise.} \end{cases}$$

.

A lower bound on the reduced cost reachable by extending label L can be computed as

$$LB(L) = \overline{c}(L) + h(L) + \frac{1}{2} \sum_{\substack{C \in \mathscr{C} \setminus \text{SFrCI} \\ i \in S(C)}} \sigma_C + \frac{1}{2} \sum_{C \in \text{SFrCI}} n_{C,i}\sigma_C. \tag{4.38}$$

The two sums aim to compensate the fact that the contribution of the under-estimated constraints $C \in \mathscr{C}$ is being considered at least 1.5 times in $\overline{c}(L)$ and $h(L)$ whenever $i \in S(C)$ or $n_{C,i} > 0$, thus tightening $LB(L)$. If a label L is such that $LB(L) > 0$, then L can be discarded. Similar fathoming rules have been implemented by Baldacci et al. [14, 15], Christofides et al. [32] and Baldacci et al. [16], for instance. Note that we have used unidirectional DP for computing functions f, g, π. From an implementation point of view, it only differs from the BDP in the fact that now all labels are inspected for extension and not only those whose load is less or equal than $Q/2$, so at the end the joining of paths is not necessary. Note also that this fathoming procedure can be generalized (and also strengthened) by keeping as resources, thus without under-estimating, the k constraints $C \in \mathscr{C}$ with the largest duals, where k is a parameter defined *a priori*. After doing a series of experiments, we let $k = \min\{20, |\mathscr{C}|/5\}$. For these constraints, the coefficients in

the sums in (4.38) can now be lifted to 1, as the contribution to the reduced cost of a customer such that $i \in S(C)$ or $n_{C,i} > 0$ is being counted twice.

Path joining As the labeling algorithm is bidirectional, the labels must be joined to construct feasible paths. Given two labels L, L' such that $v(L) = v(L')$ and $q(L) + q(L') \leq Q + d_{v(L)}$, they will produce a feasible path (one that satisfies capacity constraints and such that its reduced cost is negative) if

i. $\min\{q(L), q(L')\} \geq \frac{q(L)+q(L')-d_{v(L)}}{2}$

ii. $\max\{q(L), q(L')\} \leq \frac{q(L)+q(L')+d_{v(L)}}{2}$

iii. $v(pred(L)) < v(pred(L'))$

iv. the reduced cost of the concatenated path $P = (L, L')$ is negative.

The first two conditions are the median conditions [14] that ensure that labels L and L' are the closest possible to half of the load. The third condition ensures that if path $P = (L, L')$ is kept, then path $P' = (L', L)$ will be discarded. This way, symmetric or repeated paths will not be added to the master problem.

The dynamic programming algorithm Let us describe the labeling algorithm by means of a pseudo-code. Let L_0 be the label representing an empty path starting at the facility, such that all of the resources are set at their default values. Also, let us note that labels will be stored in buckets, and let $B(q, v)$ be the bucket storing labels L whose loads are $q(L) = q$ and such that $v(L) = v$.

Algorithm 4.1 2-cyc-SPPRC

Compute functions f, g, π using DP.
$B(0,0) \leftarrow \{L_0\}, \mathscr{V} \leftarrow \{0\}, \mathscr{R} \leftarrow \emptyset$.
repeat
  Take node v from $\mathscr{V}$ and set $\mathscr{V} \leftarrow \mathscr{V} \setminus \{v\}$.
  for $q = 0$ **to** $Q/2$ **do**
    for all $L \in B(q,v)$ such that $proc(L) =$ **false do**
      Set $proc(L) \leftarrow$ **true**.
      for all $w \in$ Neighbors of v, $w \neq 0$ **and** $q(L) + d_w \leq Q$ **and** $pred(L) \neq w$ **do**
        Create L' such that $v(L') = w$ and $pred(L') = L$. Update resources accordingly.
        Apply fathoming rule and eventually discard L'.
        Apply dominance rule and eventually discard L'.
        if L' has not been discarded **then**
          Make $B(q(L'), w) \leftarrow B(q(L'), w) \cup \{L'\}$.
          Apply dominance rule and eventually delete other labels in $B(q(L'), w)$.
          Make $\mathscr{V} \leftarrow \mathscr{V} \cup \{w\}$.
        end if
      end for
    end for
  end for
until $V = \emptyset$
Join paths $\{(L, L') : v(L) = v(L') = v, q(L) + q(L') \leq Q + d_v\}$ and fill $\mathscr{R}$
return $\mathscr{R}$

4.4.3.4 Computing lower bounds

When pricing problems are solved to optimality, it is possible to obtain a lower bound on the problem. This lower bound can then be used for fathoming the current node as well as for early termination criteria. The following proposition provides a way of computing a lower bound on the CLRP.

Proposition 4.4.1. *Let $\bar{c}_{\min}$ be the minimum reduced cost at the current iteration for columns in Ω, and let $\bar{z}$ be the value of the master problem at the current iteration. Also, let $K_{\max}$ be an upper bound on the number of vehicles that serve two or more customers.*

A valid lower bound for the CLRP is given by

$$z_{LB} = \bar{z} + K_{\max}\bar{c}_{\min}. \tag{4.39}$$

Proof Let σ be the dual variables of the linear relaxation of problem (SPF). Let $(\bar{c}_l)_{l\in\Omega}$ be the reduced costs of columns serving two or more customers, that depend on the duals σ. The Lagrangean dual of this problem, that can be written in the following form, provides a valid lower bound for the CLRP

$$L(\sigma) = \bar{z} + \min\{\sum_{l\in\Omega} \bar{c}_l\lambda_l : \sum_{l\in\Omega} \lambda_l \leq K_{\max}\}. \tag{4.40}$$

But now, as $\bar{c}_{\min} \leq 0$ then $\min\{\sum_{l\in\Omega} \bar{c}_l\lambda_l : \sum_{l\in\Omega} \lambda_l \leq K_{\max}\} \leq K_{\max}\bar{c}_{\min}$. □

For every candidate set I' we use $K_{\max} = K_m(I')$ as described in the first bounding procedure.

4.4.4 Enumeration of remaining columns

For each subset of facilities I' as obtained after the first bounding procedure and not discarded after the second procedure, let z_{LB} and σ be the lower bound at the end of the second bounding procedure and the dual variables associated to such lower bound. If procedure ENUM-ESPPRC was successful to generate the column set $\mathscr{P}$, we simply compute the reduced cost of columns in $\mathscr{P}$ and add to the master problem those columns l such that $\bar{c}_l < z_{UB} - z_{LB}$. We then solve the resulting integer problem using a general-purpose solver such as CPLEX. If, however, we were not able to obtain set $\mathscr{P}$, we first check whether the upper bound z_{UB} improved during the second bounding procedure after the consideration of set I'. In this case, we run again algorithm ENUM-ESPPRC but now with the updated upper bound, as the performance of algorithm ENUM-ESPPRC depends strongly on the gap $z_{UB} - z_{LB}$. Otherwise, we start the following procedure with the hope of getting a better upper bound (if any), and in the worst case it gives us a method for tightening the gap.

i. Let $\Delta \leftarrow (z_{UB} - z_{LB})/10$. Set $k \leftarrow 1$.

ii. Let $z'_{UB} \leftarrow z_{LB} + k\Delta$ and try to generate all of the columns whose reduced costs are smaller or equal than $k\Delta$. If more than $\Delta_{max} = 10^6$ columns are found or if we run out of memory, we exit. Otherwise we go to step (iii).

iii. Solve the resulting integer problem to optimality. If a new upper bound was found with value $z^* < z_{UB}$, set $z_{UB} \leftarrow z^*$ and $z_{LB} \leftarrow \min\{z_{UB}, z'_{UB}\}$. If $z_{LB} = z'_{UB}$ then exit. Otherwise, if either $z'_{UB} < z^*$ or the problem was solved to optimality but no integer solution was found with value less than z_{UB}, set $z_{LB} = z'_{UB}$. If $k < 10$ do $k \leftarrow k + 1$ and go back to (ii).

This method generalizes the one proposed by Baldacci et al. [16] by artificially lowering the optimality gap and iteratively increasing it, thus reducing the negative impact of an initial upper bound of poor quality. Let us describe the algorithm for solving the column enumeration problem. This algorithm is a variation of the Elementary SPPRC (ESPPRC) and we call it ENUM-ESPPRC.

4.4.4.1 The column enumeration algorithm

Algorithm ENUM-ESPPRC is based on the solution of the ESPPRC, and so as the 2-cyc-SPPRC, is solved by means of bidirectional dynamic programming. The method presented in this paper differs from the one proposed by Baldacci et al. [16] mainly in the fathoming rule that considers the inclusion of the strong constraints in the value of the completion bound for a given path label. As for the description of the 2-cyc-SPPRC, we first describe the definition of a label in the recursion of the dynamic programming algorithm. Then, we describe the dominance rules used to discard labels. After that, we describe a fathoming rule that can be applied in order to also discard labels that cannot lead to a column of reduced cost smaller than a desired threshold. Next, we describe the path joining procedure to build feasible paths from a given pair of labels. At the end, we describe the skeleton of the algorithm.

Label definition We define a label L containing the same information as for the 2-cyc-SPPRC algorithm plus

i. A cost $c(L)$ representing the cost of the path represented by label L.

ii. Additional resources associated to nodes. For every customer $j \in J$ we associate a boolean variable $res_j(L)$ equal to $true$ if $j \in V(L)$, 0 otherwise.

Dominance rule Given two labels L, L', we say that L is dominated by L' if

i. $v(L) = v(L')$

ii. $V(L) = V(L')$

iii. $c(L) \geq c(L')$

Now, dominance is done with respect to the costs instead of the reduced costs. A Label L that is found to be dominated by another label L' is removed, and recursively also all of its successors.

Fathoming rule A similar fathoming rule as the one used for the 2-cyc-SPPRC can be applied. Indeed, it only differs from the one used for the 2-cyc-SPPRC in the parameter k for the number of non under-estimated constraints that is set to $k = |\mathscr{C}|$. Thus, a lower bound $LB(L)$ on the reduced cost of a label L after extending it is given by

$$LB(L) = \overline{c}(L) + h(L) + \sum_{\substack{C \in \mathscr{C} \setminus \text{SFrCI} \\ i \in S(C)}} \sigma_C + \sum_{C \in \text{SFrCI}} n_{C,i}\sigma_C, \tag{4.41}$$

where $h(L)$, σ_C and $n_{C,i}$ are as defined for the fathoming rule of the 2-cyc-SPPRC. Now, a label L will be discarded if $LB(L) \geq z_{UB} - z_{LB}$.

Path joining A similar joining procedure can be applied to algorithm ENUM-ESPPRC as with the 2-cyc-SPPRC, with the main difference that now cycles are not allowed at all. Given two labels L, L' such that $v(L) = v(L')$ and $q(L) + q(L') \leq Q + d_{v(L)}$, they will produce a feasible path (one that satisfies capacity constraints and such that its reduced cost is smaller than the desired threshold) if

i. $\min\{q(L), q(L')\} \geq \frac{q(L)+q(L')-d_{v(L)}}{2}$

ii. $\max\{q(L), q(L')\} \leq \frac{q(L)+q(L')+d_{v(L)}}{2}$

iii. $v(pred(L)) < v(pred(L'))$

iv. $V(L) \cap V(L') = \{0, v(L)\}$

v. the reduced cost of the concatenated path $P = (L, L')$ is smaller than $z_{UB} - z_{LB}$.

Now, condition (iv) ensures that paths L, L' only share the facility and the joining node.

The dynamic programming algorithm Let us describe the labeling algorithm by means of a pseudo-code. Just as before, label L_0 represents an empty path starting at the facility, such that all of the resources are set at their default values. Labels will be stored in buckets, and let $B(q, v)$ be the bucket storing labels L whose loads are $q(L) = q$ and such that $v(L) = v$.

Algorithm 4.2 ENUM-ESPPRC

Compute functions f, g, π using DP.
$B(0,0) \leftarrow \{L_0\}, \mathscr{V} \leftarrow \{0\}, \mathscr{R} \leftarrow \emptyset$
repeat
  Take node v from $\mathscr{V}$ and set $\mathscr{V} \leftarrow \mathscr{V} \setminus \{v\}$
  for $q = 0$ **to** $Q/2$ **do**
    for all $L \in B(q, v)$ such that $proc(L) =$ **false do**
      Set $proc(L) \leftarrow$ **true**.
      for all $w \in$ Neighbors of v, $w \neq 0$ **and** $q(L) + d_w \leq Q$ **and** $w \notin V(L)$ **do**
        Create L' such that $v(L') = w$ and $pred(L') = L$. Update resources accordingly.
        Apply fathoming rule and eventually discard L'.
        Apply dominance rule and eventually discard L'.
        if L' has not been discarded **then**
          Make $B(q(L'), w) \leftarrow B(q(L'), w) \cup \{L'\}$.
          Apply dominance rule and eventually delete other labels in $B(q(L'), w)$.
          Make $\mathscr{V} \leftarrow \mathscr{V} \cup \{w\}$.
        end if
      end for
    end for
  end for
until $V = \emptyset$
Join paths $\{(L, L') : v(L) = v(L') = v, q(L) + q(L') \leq Q + d_v\}$ and fill $\mathscr{R}$.
return $\mathscr{R}$

4.4.5 Computational issues

We now make some observations that can help to accelerate the algorithm.

4.4.5.1 Initial set of columns

An initial set of columns is required in column generation algorithms. Indeed, at every iteration of the CG, a feasible solution of the master problem is needed for running the pricing algorithms. In our algorithm, we let the initial set of columns contain only the single-customer variables y. Additionally, we also add slack and artificial variables to the formulation so the problem will always have a feasible solution.

4.4.5.2 Stabilization of the column generation

With the aim of reducing the oscillation of the dual variables during the first iterations of the column generation process, we use a box-pen method [53] for stabilizing the duals of the degree constraints (4.12). For every set $I' \in \mathscr{I}$, the centers are initially set to the optimal dual variables of the degree constraints (4.1) after performing the first bounding procedure.

4.4.5.3 Column pool management

For some instances, the quantity of columns added can be huge and, moreover, most of them will be useless. In fact, it is known that at the beginning of the column generation process, many columns are generated that soon will become non-basic for the rest of the algorithm. We keep a pool of columns and keep track of the number of consecutive iterations that columns have been non-basic. Every 30 iterations we check and delete all columns having been inactive for more than 30 iterations. Note that after the creation of set $\mathscr{P}$ during the second bounding procedure, the columns deleted from the problem must be inserted back into $\mathscr{P}$.

4.4.5.4 Memory management

The dynamic programming algorithms can be very demanding in terms of memory. In fact, every new created label needs to be allocated in memory. In this context, the *new* and *delete* operators of C++ (or *malloc* and *free* operators in the case of C) can be very inefficient. We have decided to manage our own memory pool, in which dynamic memory is allocated in chunks of 400 MB. The newly created labels are thus allocated inside the previously allocated memory.

4.5 Computational Experience

We have run our method on an Intel Xeon E5462, 3.0 Ghz processor with 16GB of memory. The code was compiled with the Intel C++ compiler v11.0 and executed on Linux, kernel 2.6. Linear and integer programs were solved by CPLEX 12.2. The pricing algorithms 2-cyc-SPPRC and ENUM-ESPPRC have been coded in C++ using the same compiler as before. The algorithm has been tested over five sets of instances from the literature, containing in total 71 instances. The first family ($\mathscr{F}_1$) has been adapted by Barreto [18] from other vehicle routing problems in the literature and contains 16 instances with capacitated vehicles and facilities. The second set of instances ($\mathscr{F}_2$) has been developed by Prodhon [123] and contains 30 instances with capacitated vehicles and facilities. The third set of instances ($\mathscr{F}_3$) has been introduced by Akca et al. [4] and contains 12 instances with capacitated vehicles and facilities. The fourth set of instances ($\mathscr{F}_4$) has been introduced by Tuzun and Burke [142] and contains 9 instances with capacitated vehicles and uncapacitated facilities. The fifth and last set of instances ($\mathscr{F}_5$) has been introduced by Baldacci et al. [16] and contains 4 instances with capacitated vehicles and uncapacitated facilities. The dimensions of the instances vary from very small instances with 12 customers and 2 facilities up to very large instances with 199 customers and 14 facilities. We compare our results against those obtained by other exact algorithms, namely the methods of Belenguer et al. [19], Contardo et al. [36] and Baldacci et al. [16]. We use as upper bound the best solution available in the literature for every instance. In Tables 4.I-4.V we present the detailed results obtained by our al-

gorithm for every instance and for each of the three bounding procedures. The columns in these tables are as follows:

i. *Instance*: name of the instance.

ii. z_{UB}: objective function value of the best feasible solution available in the literature.

iii. z^*: objective function value of the best feasible solution found by our algorithm. The text in bold characters indicates that this value is strictly lower than the one in column labeled z_{UB}.

iv. gap_1, t_1: gap obtained and CPU time taken by the first bounding procedure. The gap is computed as follows: $(z^* - z_{LB_1})/z^* \times 100$.

v. gap_2, t_2: gap obtained and CPU time taken by the second bounding procedure.

vi. gap_3, t_3: gap obtained and CPU time taken by the column enumeration procedure.

vii. $|\mathscr{I}|$: number of subsets obtained by the first bounding procedure.

viii. $|\mathscr{R}_{1,2}|$: maximum number of columns found by the procedure ENUM-ESPPRC after the second bounding procedure and the final enumeration step, respectively. This maximum is taken over all subsets $I' \subseteq \mathscr{I}$.

ix. t: overall CPU time.

As shown in these tables, our algorithm is capable of solving 58 out of the 71 instances considered. Moreover, all instances of families $\mathscr{F}_1$ and $\mathscr{F}_3$ (28 in total) are solved to optimality, and for none of them was procedure ENUM-ESPPRC called during the third bounding procedure. Finally, instances Chr-75x10ba, ppw-50x5-2b, ppw-100x5-2b and ppw-200x10-3a were solved to optimality for the first time, and we have improved the best feasible solution for three more instances (ppw-100x5-0b, P113112 and P131112). As a matter of fact, our method is able to solve all instances with 85 customers or less.

We first compare our method against the branch-and-cut algorithms of Belenguer et al. [19] and of Contardo et al. [36]. In Tables 4.VI-4.VIII we establish the gaps and CPU times obtained by every algorithm on three sets of instances. In these tables, headers BBPPW, CCG-BC and CCG-BCP stand for the methods of Belenguer et al. [19],

Contardo et al. [36] and this work, respectively. In the case of method CCG-BC we consider the branch-and-cut algorithm with the two-index vehicle-flow formulation of the problem. In the case of the branch-and-cut algorithms, columns labeled gap_{lr}, t_{lr}, gap and t stand for the gaps and CPU times for the root node relaxation and after the whole branching tree (with a maximum CPU time of 2 hours). In the case of method CCG-BCP, columns gap_1, t_1 stand for the gap obtained after the first bounding procedure, and columns labeled gap, t stand for the final gap and the total CPU time spent by the method. We highlight in bold characters whenever a method dominates the other two in terms of bound quality. First of all, the first bounding procedure produces better bounds than the flow-based algorithms at the root node. This is not surprising since this procedure uses the code of CCG-BC for doing a partial branch-and-bound on the location variables. At the end, our method is able to produce tighter gaps than the other two. Although a CPU-based comparison can be difficult (because each algorithm was run on different machines), it is worth noting that our method was some orders of magnitude faster on the instances of family $\mathscr{F}_3$ (Table 4.VIII). Moreover, we can solve 48 of the considered instances, 20 more than BBPPW and 18 more than CCG-BC.

Finally, we compare the proposed methodology against the column generation method of Baldacci et al. [16]. In Tables 4.IX-4.XII we compare the three bounding procedures introduced in this paper against the similar bounds used in the method of Baldacci et al. [16]. Note that, although the second and third bounding procedures in both methods are very similar, the first bounding used by Baldacci et al. [16] is a relaxation of the set-partitioning formulation, while in our case it is based on the two-index vehicle-flow formulation of the problem. In these tables the legend is analogous to that used for the previous set of tables. We also highlight in bold characters whenever a bound dominates the other. As shown in these tables, our method is able to produce tighter bounds than that of Baldacci et al. [16] for most instances and for every bounding procedure. Our first bounding procedure is quite effective whenever branching decisions on the location variables have a significant impact on either the bounds or the feasibility of the problem. Indeed, this is the case for all sets of instances except for $\mathscr{F}_5$. Our first bounding procedure obtains smaller gaps than that of Baldacci et al. [16] in 51 out of the 71 instances

considered. For the second bounding procedure, our algorithm obtains smaller gaps in 43 out of the 71 instances. This shows the strength of the set-partitioning formulation with the additional cuts. Our third bounding procedure, although it can be very time consuming, is shown to be effective for solving instances Chr-75x10ba and ppw-200x10-3a in which the initial upper bounds are significantly improved during this procedure. In general, our algorithm is able to solve four instances that are not solved by the method of Baldacci et al. [16], and improves the best known feasible solution in three other instances. However, for the instances in family $\mathscr{F}_5$ our method is outperformed by that of Baldacci et al. [16]. The overall results suggest that our method is competitive against the one of Baldacci et al. [16]. This is the result of several refinements with respect to their method, namely the use of the new cuts, as well as the use of efficient pricing algorithms that properly handle these new cuts. This includes the use of stronger fathoming procedures based on the solution of a 2-cyc-SPPRC with resources.

4.6 Concluding remarks

In this paper, we have presented an exact method for solving the CLRP. The methodology consists in formulating the CLRP as a set-partitioning problem that is solved in three stages: in a first stage we consider the two-index formulation and branch on the location variables. This strategy works well for instances in which branching decisions on the location variables have a significant impact on the feasibility or the bound at the resulting nodes in the branching tree. The remaining gap is then closed by sequentially applying two procedures, both based on the set-partitioning formulation and solved by means of column-and-cut generation. The algorithm proposed in this paper is able to produce the tightest gaps on a large number of instances. In addition, it has solved to optimality four previously open instances and improved the best known feasible solution for three additional ones. The methodology can be easily adapted to solve other routing problems. For instance, it would be interesting to measure the impact of y-SCC and SFrCI cuts on solving hard instances of the CVRP. With respect to the pricing algorithm introduced in this paper, the consideration of SDEG cuts allows to get lower bounds

that are comparable to those obtained when pricing on elementary routes in a fraction of the computational effort. Indeed, in most cases only a fraction of SDEG cuts need to be added to the master problem to obtain significant improvements in the lower bound. Moreover, we show how to take advantage of this pricing problem in the computation of tight fathoming rules that speed up the whole algorithm. Further research related to the methodology introduced in this paper should address the development of new cutting planes for the set-partitioning formulation and to adapt some of them to other routing problems.

Acknowledgements

The authors would like to thank Simon Spoorendonk and Enrico Bartolini for their insightful comments. Thanks are also due to the *Natural Sciences and Engineering Research Council of Canada* (NSERC) and *Le fonds québécois de la recherche sur la nature et les technologies* (FQRNT) for their financial support.

Instance	z_{UB}	z^*	gap_1	$\lvert\mathscr{I}\rvert$	t_1	gap_2	$\lvert\mathscr{R}_1\rvert$	t_2	gap_3	$\lvert\mathscr{R}_2\rvert$	t_3	t
Perl83-12x2	204.00	204.00*	0.00	1	0.02	0.00	4	0.01	0.00	0	0.00	0.03
Gas67-21x5	424.90	424.90*	1.61	2	0.25	0.00	17	0.08	0.00	0	0.00	0.33
Gas67-22x5	585.11	585.11*	0.10	1	0.05	0.00	24	0.18	0.00	0	0.00	0.23
Min92-27x5	3062.02	3062.02*	0.00	1	0.21	0.00	15	0.10	0.00	0	0.00	0.31
Gas67-29x5	512.10	512.10*	1.88	1	0.44	0.00	2077	3.76	0.00	0	0.00	4.20
Gas67-32x5	562.22	562.22*	1.24	1	0.57	0.00	12512	5.95	0.00	0	0.00	6.52
Gas67-32x5-2	504.33	504.33*	0.01	1	0.51	0.00	9	0.19	0.00	0	0.00	0.70
Gas67-36x5	460.37	460.37*	0.00	0	1.04	0.00	0	0.00	0.00	0	0.00	1.04
Chr69-50x5ba[1]	565.62	565.62*	1.58	2	6.53	0.00	10797	5.55	0.00	0	0.00	12.08
Chr69-50x5be[2]	565.60	565.60*	2.14	5	8.70	0.00	11928	18.66	0.00	0	0.00	27.36
Perl83-55x15	1112.06	1112.06	1.96	200	197.21	0.00	26860	43.54	0.00	0	0.00	240.75
Chr69-75x10ba[1]	886.30	**844.40***	7.27	489	1243.61	0.48	2199569	5749.77	0.00	0	13.04	7006.42
Chr69-75x10be[2]	848.85	848.85*	6.71	195	485.76	0.00	442577	3945.23	0.00	0	0.00	4430.99
Chr69-75x10bmw[3]	802.08	802.08*	6.02	117	207.56	0.00	1172461	1733.37	0.00	0	0.00	1940.93
Perl83-85x7	1622.50	1622.50*	1.65	19	76.12	0.00	767712	101.80	0.00	0	0.00	177.92
Chr69-100x10	833.43	833.43*	1.81	27	419.35	0.00	771623	1130.42	0.00	0	0.00	1549.77
Average			2.12		165.50	0.03		796.16	0.00		0.81	962.47

[1] Instance used by Barreto [18].
[2] Instance used by Belenguer et al. [19].
[3] Instance used by Baldacci et al. [16].
* Optimal solution.

Table 4.I: Results on family $\mathscr{F}_1$

Instance	z_{UB}	z^*	gap_1	$\lvert\mathscr{I}\rvert$	t_1	gap_2	$\lvert\mathscr{R}_1\rvert$	t_2	gap_3	$\lvert\mathscr{R}_2\rvert$	t_3	t
ppw-20x5-0a	54793	54793*	3.06	3	0.29	0.00	372	0.31	0.00	0	0.00	0.60
ppw-20x5-0b	39104	39104*	0.00	0	0.03	0.00	0	0.00	0.00	0	0.00	0.03
ppw-20x5-2a	48908	48908*	2.40	2	0.14	0.00	587	0.42	0.00	0	0.00	0.56
ppw-20x5-2b	37542	37542*	0.00	0	0.02	0.00	0	0.00	0.00	0	0.00	0.02
ppw-50x5-0a	90111	90111*	5.95	3	7.25	0.15	51513	9.69	0.00	0	0.14	17.08
ppw-50x5-0b	63242	63242*	3.55	1	4.18	0.58	2456549	296.71	0.00	0	177.14	478.03
ppw-50x5-2a	88298	88298*	3.92	3	6.31	0.00	21673	7.18	0.00	0	0.00	13.49
ppw-50x5-2b	67340	**67308***	3.77	3	2.84	0.00	346704	441.49	0.00	0	0.00	444.33
ppw-50x5-2a'	84055	84055*	1.99	2	7.44	0.03	134118	22.01	0.00	0	0.08	29.53
ppw-50x5-2b'	51822	51822*	0.69	2	1.49	0.00	40913	5.76	0.00	0	0.00	7.25
ppw-50x5-3a	86203	86203*	3.93	3	13.19	0.82	113809	25.12	0.00	0	55.93	94.24
ppw-50x5-3b	61830	61830*	2.30	4	5.78	0.00	937460	80.95	0.00	0	0.00	86.73
ppw-100x5-0a	274814	274814*	4.55	4	144.98	0.20	469031	238.71	0.00	0	62.44	446.13
ppw-100x5-0b	214392	**213568**	3.14	2	119.05	0.46	Δ_{max}	24902.90	0.29	248557	22824.30	47846.20
ppw-100x5-2a	193671	193671*	3.52	1	36.06	0.06	64646	19.92	0.00	0	0.55	56.53
ppw-100x5-2b	157173	**157095***	2.14	2	42.68	0.10	978507	10486.60	0.00	0	2047.04	12576.30
ppw-100x5-3a	200079	200079*	3.55	1	35.56	0.19	529311	53.88	0.00	0	34.30	123.74
ppw-100x5-3b	152441	152441*	1.95	1	23.85	0.00	210300	446.04	0.00	0	0.00	469.89
ppw-100x10-0a	289017	289017	5.77	5	1147.03	1.70	Δ_{max}	1623.27	1.27	251056	43573.30	46343.60
ppw-100x10-0b	234641	234641	4.44	5	163.75	2.06	Δ_{max}	15156.30	1.94	321769	22772.60	38092.70
ppw-100x10-2a	243590	243590*	2.82	8	169.38	0.32	1811787	729.95	0.00	0	1353.58	2252.91
ppw-100x10-2b	203988	203988*	0.79	3	72.55	0.00	161828	252.17	0.00	0	0.00	324.72
ppw-100x10-3a	252421	252421	6.02	18	1425.99	2.00	Δ_{max}	1026.68	1.60	166861	22460.70	24913.40
ppw-100x10-3b	204597	204597	4.02	6	150.36	1.49	Δ_{max}	50030.40	1.36	151231	30015.40	80196.10
ppw-200x10-0a	479425	479425	8.66	31	3861.00	1.28	Δ_{max}	13040.60	1.15	894085	32897.40	49799.00
ppw-200x10-0b	378773	378773	5.32	10	3367.07	1.19	Δ_{max}	148669.00	1.19	Δ_{max}	85063.50	237100.00
ppw-200x10-2a	450468	450468	5.06	3	359.83	0.88	Δ_{max}	8838.32	0.80	683782	22043.00	31241.20
ppw-200x10-2b	374435	374435	3.10	3	566.81	0.42	Δ_{max}	61771.20	0.37	Δ_{max}	30157.70	92495.60
ppw-200x10-3a	472898	**469433***	6.61	16	3788.52	0.12	Δ_{max}	10313.30	0.00	169099	4636.90	18738.80
ppw-200x10-3b	364178	364178	4.92	6	2482.11	1.00	Δ_{max}	37910.00	1.00	Δ_{max}	4621.82	45013.90
Average			3.60		600.18	0.50		12879.96	0.37		10826.59	24306.75
Average on solved instances			2.87		218.13	0.13		1171.51	0.00		418.41	1808.05
Average on unsolved instances			5.04		1364.30	1.25		36296.87	1.10		31642.97	69304.17

* Optimal solution.

Table 4.II: Results on family $\mathscr{F}_2$

Instance	z_{UB}	z^*	gap_1	$\lvert\mathscr{I}\rvert$	t_1	gap_2	$\lvert\mathscr{R}_1\rvert$	t_2	gap_3	$\lvert\mathscr{R}_2\rvert$	t_3	t
cr30x5a-1	819.52	819.52*	2.89	2	0.60	0.00	2891	1.85	0.00	0	0.00	2.45
cr30x5a-2	821.50	821.50*	3.73	1	0.38	0.00	857	3.34	0.00	0	0.00	3.72
cr30x5a-3	702.30	702.30*	0.00	1	0.44	0.00	14	0.06	0.00	0	0.00	0.50
cr30x5b-1	880.02	880.02*	2.69	2	1.01	0.00	2963	3.56	0.00	0	0.00	4.57
cr30x5b-2	825.32	825.32*	1.22	1	0.97	0.00	29	0.27	0.00	0	0.00	1.24
cr30x5b-3	884.60	884.60*	2.33	1	0.92	0.00	31	0.31	0.00	0	0.00	1.23
cr40x5a-1	928.10	928.10*	3.30	7	3.99	0.00	14135	10.68	0.00	0	0.00	14.67
cr40x5a-2	888.42	888.42*	2.80	3	2.25	0.19	53615	9.58	0.00	0	0.05	11.88
cr40x5a-3	947.30	947.30*	3.02	4	3.09	0.00	9166	8.27	0.00	0	0.00	11.36
cr40x5b-1	1052.04	1052.04*	5.78	8	5.49	0.00	2522	5.00	0.00	0	0.00	10.49
cr40x5b-2	981.54	981.54*	2.09	3	1.63	0.00	329	2.14	0.00	0	0.00	3.77
cr40x5b-3	964.33	964.33*	2.00	1	1.67	0.00	33	1.01	0.00	0	0.00	2.68
Average			2.65		1.87	0.02		3.84	0.00		0.00	5.71

* Optimal solution.

Table 4.III: Results on family $\mathscr{F}_3$

Instance	z_{UB}	z^*	gap_1	$\lvert\mathscr{I}\rvert$	t_1	gap_2	$\lvert\mathscr{R}_1\rvert$	t_2	gap_3	$\lvert\mathscr{R}_2\rvert$	t_3	t
P111112	1467.68	1467.68*	4.63	42	885.82	0.00	2065379	3581.25	0.00	0	0.00	4467.07
P111212	1394.80	1394.80*	4.67	52	859.13	0.00	3429771	20689.10	0.00	0	0.00	21548.30
P112112	1167.16	1167.16*	2.74	1	75.47	0.00	35633	235.30	0.00	0	0.00	310.77
P112212	791.66	791.66*	1.36	1	21.32	0.00	4800861	1818.15	0.00	0	0.00	1839.47
P113112	1245.45	**1238.24**	3.95	13	454.58	0.61	Δ_{max}	118970.00	0.34	Δ_{max}	52278.40	171703.00
P113212	902.26	902.26*	0.49	2	51.30	0.00	8502	101.93	0.00	0	0.00	153.23
P131112	1900.70	**1896.98**	6.24	206	11926.90	0.85	Δ_{max}	299823.00	0.64	Δ_{max}	38047.20	349797.00
P131212	1965.12	1965.12	8.43	282	10682.50	0.95	Δ_{max}	207942.00	0.82	258029	33676.30	252301.00
P132112	1443.33	1443.32*	5.68	16	3582.12	0.00	1120735	15021.90	0.00	0	0.00	18604.00
Average			4.24		3171.02	0.27		74242.51	0.20		13777.99	91191.54
Average on solved instances			3.26		912.53	0.00		6907.94	0.00		0.00	7820.47
Average on unsolved instances			6.21		7687.99	0.80		208911.67	0.60		41333.97	257933.67

* Optimal solution.

Table 4.IV: Results on family $\mathscr{F}_4$

Instance	z_{UB}	z^*	gap_1	$\lvert\mathscr{I}\rvert$	t_1	gap_2	$\lvert\mathscr{R}_1\rvert$	t_2	gap_3	$\lvert\mathscr{R}_2\rvert$	t_3	t
M-n150x14a	1352.93	1352.93*	9.61	2331	34342.60	0.09	3251152	351540.00	0.00	0	1.32	385884.00
M-n150x14b	1212.46	1212.46*	7.50	2465	39786.80	0.20	4507242	352454.00	0.00	0	30.68	392272.00
M-n199x14a	1644.35	1644.35*	12.06	1598	64412.40	0.15	4557968	616522.00	0.00	0	10.81	680945.00
M-n199x14b	1480.43	1480.43*	10.22	1513	67992.50	0.09	2141465	1006620.00	0.00	0	1.98	1074610.00
Average			9.85		51633.57	0.13		581784.00	0.00		11.20	633427.75

* Optimal solution.

Table 4.V: Results on family $\mathscr{F}_5$

Instance	z_{UB}	z^*	BBPPW				CCG-BC				CCG-BCP			
			gap_{lr}	t_{lr}	gap	t	gap_{lr}	t_{lr}	gap	t	gap_1	t_1	gap	t
Perl83-12x2	204.00	203.98					0.61	0.01	0.00	0.02	**0.00**	0.02	0.00	0.03
Gas67-21x5	424.90	424.90	3.91	0.22	0.00	0.60	3.99	0.21	0.00	0.59	**1.61**	0.25	0.00	0.33
Gas67-22x5	585.11	585.11	0.28	0.14	0.00	0.20	0.10	0.04	0.00	0.07	**0.10**	0.05	0.00	0.23
Min92-27x5	3062.02	3062.02	5.62	0.27	0.00	0.80	5.62	0.29	0.00	0.73	**0.00**	0.21	0.00	0.31
Gas67-29x5	512.10	512.10	4.72	0.41	0.00	1.00	4.89	0.46	0.00	1.01	**1.88**	0.44	0.00	4.20
Gas67-32x5	562.22	562.22	6.11	0.61	0.00	3.45	5.72	0.51	0.00	1.76	**1.24**	0.57	0.00	6.52
Gas67-32x5-2	504.33	504.33	3.46	0.39	0.00	0.50	3.27	0.80	0.00	1.01	**0.01**	0.51	0.00	0.70
Gas67-36x5	460.37	460.37	2.79	0.72	0.00	2.10	1.30	1.40	0.00	2.80	**0.00**	1.04	0.00	1.04
Chr69-50x5ba	565.62	565.62					5.62	3.74	0.00	44.78	**1.58**	6.53	0.00	12.08
Chr69-50x5be	565.60	565.60	10.15	2.70	0.00	181.10	8.85	3.04	0.00	68.79	**2.14**	8.70	0.00	27.36
Perl83-55x15	1112.06	1112.06					3.42	6.17	0.70	7496.92	**1.96**	197.21	**0.00**	240.75
Chr69-75x10ba	886.30	**844.40**					10.23	23.54	4.51	7408.62	**7.27**	1243.61	**0.00**	7006.42
Chr69-75x10be	848.85	848.85	10.83	37.66	4.48	3017.83	10.42	15.25	3.26	7367.52	**6.71**	485.76	**0.00**	4430.99
Chr69-75x10bmw	802.08	802.08					9.27	20.18	3.37	7468.83	**6.02**	207.56	**0.00**	1940.93
Perl83-85x7	1622.50	1622.50					2.53	18.93	0.68	7557.62	**1.65**	76.12	**0.00**	177.92
Chr69-100x10	833.43	833.43					4.89	9.71	0.51	7385.58	**1.81**	419.35	**0.00**	1549.77
Average BBPPW[1]			5.32	4.79	0.50	356.40	4.91	2.44	0.36	827.14	**1.52**	55.28	**0.00**	496.85
Average CCG-BC[2]							5.05	6.52	0.81	2800.42	**2.12**	165.50	**0.00**	962.47

[1] Average on instances reported by Belenguer et al. [19].

[2] Average on instances reported by Contardo et al. [36].

Table 4.VI: Comparison with the methods of Belenguer et al. [19] and Contardo et al. [36] on family $\mathscr{F}_1$

Instance	z_{UB}	z^*	BBPPW				CCG-BC				CCG-BCP			
			gap_{lr}	t_{lr}	gap	t	gap_{lr}	t_{lr}	gap	t	gap_1	t_1	gap	t
ppw-20x5-0a	54793	54793	7.13	0.34	0.00	2.41	4.57	0.35	0.00	5.04	**3.06**	0.29	0.00	0.60
ppw-20x5-0b	39104	39104	0.00	0.17	0.00	0.13	0.00	0.02	0.00	0.03	0.00	0.03	0.00	0.03
ppw-20x5-2a	48908	48908	3.53	0.25	0.00	2.81	2.71	0.26	0.00	1.31	**2.40**	0.14	0.00	0.56
ppw-20x5-2b	37542	37542	0.00	0.06	0.00	0.06	0.00	0.01	0.00	0.02	0.00	0.02	0.00	0.02
ppw-50x5-0a	90111	90111	11.61	12.38	2.26	7212.25	10.94	27.10	1.77	7336.51	**5.95**	7.25	**0.00**	17.08
ppw-50x5-0b	63242	63242	7.96	3.97	1.37	5557.95	7.50	5.05	1.23	7304.54	**3.55**	4.18	**0.00**	478.03
ppw-50x5-2a	88298	88298	7.48	8.14	1.11	7208.06	7.52	5.08	1.00	7265.57	**3.92**	6.31	**0.00**	13.49
ppw-50x5-2b	67340	**67308**	5.19	2.45	1.43	6013.58	5.63	2.75	1.17	7282.73	**3.77**	2.84	**0.00**	444.33
ppw-50x5-2a	84055	84055	1.97	7.19	0.47	7207.95	**1.95**	29.50	0.28	7244.32	1.99	7.44	**0.00**	29.53
ppw-50x5-2b	51822	51822	**0.67**	1.55	0.00	9.16	0.86	1.76	0.00	10.64	0.69	1.49	0.00	7.25
ppw-50x5-3a	86203	86203	11.25	6.88	1.80	7206.95	10.23	14.67	1.16	7283.89	**3.93**	13.19	**0.00**	94.24
ppw-50x5-3b	61830	61830	7.95	3.14	0.00	96.86	6.26	4.38	0.00	71.76	**2.30**	5.78	0.00	86.73
ppw-100x5-0a	274814	274814					**3.56**	2509.03	2.36	7293.80	4.55	144.98	**0.00**	446.13
ppw-100x5-0b	214392	**213568**					3.21	391.48	2.19	7420.00	**3.14**	119.05	**0.29**	47846.20
ppw-100x5-2a	193671	193671					3.77	365.93	1.60	7398.86	**3.52**	36.06	**0.00**	56.53
ppw-100x5-2b	157173	**157095**					2.34	83.27	0.78	7323.10	**2.14**	42.68	**0.00**	12576.30
ppw-100x5-3a	200079	200079					8.82	108.07	1.44	7340.95	**3.55**	35.56	**0.00**	123.74
ppw-100x5-3b	152441	152441					5.08	27.40	0.57	7332.99	**1.95**	23.85	**0.00**	469.89
ppw-100x10-0a	289017	289017					7.88	1133.84	3.74	7394.33	**5.77**	1147.03	**1.27**	46343.60
ppw-100x10-0b	234641	234641					4.74	147.20	2.48	7334.20	**4.44**	163.75	**1.94**	38092.70
ppw-100x10-2a	243590	243590					4.07	1473.84	1.40	7351.82	**2.82**	169.38	**0.00**	2252.91
ppw-100x10-2b	203988	203988					2.50	90.42	0.00	4734.83	**0.79**	72.55	0.00	324.72
ppw-100x10-3a	252421	252421					8.65	740.38	4.02	7326.18	**6.02**	1425.99	**1.60**	24913.40
ppw-100x10-3b	204597	204597					5.00	112.22	2.15	7338.32	**4.02**	150.36	**1.36**	80196.10
ppw-200x10-1a	479425	478845									**8.66**	3861.00	**1.15**	49799.00
ppw-200x10-1b	378773	378773									**5.32**	3367.07	**1.19**	237100.00
ppw-200x10-2a	450468	450468									**5.06**	359.83	**0.80**	31241.20
ppw-200x10-2b	374435	374435									**3.10**	566.81	**0.37**	92495.60
ppw-200x10-3a	472898	**469433**									**6.61**	3788.52	**0.00**	18738.80
ppw-200x10-3b	364178	364178									**4.92**	2482.11	**1.00**	45013.90
Average BBPPW[1]			5.39	3.88	0.70	3376.51	4.85	7.58	0.55	3650.53	**2.63**	4.08	**0.00**	97.66
Average CCG-BC[2]							4.91	303.08	1.22	5391.49	**3.09**	149.17	**0.27**	10617.25

[1] Average on instances reported by Belenguer et al. [19].
[2] Average on instances reported by Contardo et al. [36].

Table 4.VII: Comparison with the methods of Belenguer et al. [19] and Contardo et al. [36] on family $\mathscr{F}_2$

Instance	z_{UB}	z^*	BBPPW				CCG-BC				CCG-BCP			
			gap_{lr}	t_{lr}	gap	t	gap_{lr}	t_{lr}	gap	t	gap_1	t_1	gap	t
r30x5a-1	819.51	819.51	4.28	0.70	0.00	50.22	3.33	0.89	0.00	3.23	**2.89**	0.60	0.00	2.45
r30x5a-2	821.50	821.46	6.41	0.53	0.00	53.89	5.89	0.41	0.00	8.77	**3.73**	0.38	0.00	3.72
r30x5a-3	702.30	702.29	1.09	0.52	0.00	0.73	0.56	0.71	0.00	0.91	**0.00**	0.44	0.00	0.50
r30x5b-1	880.02	880.02	7.58	0.47	0.00	8.48	7.39	0.52	0.00	9.05	**2.69**	1.01	0.00	4.57
r30x5b-2	825.30	825.30	4.38	0.50	0.00	1.09	3.52	1.31	0.00	2.55	**1.22**	0.97	0.00	1.24
r30x5b-3	884.60	884.58	3.14	0.95	0.00	5.63	3.33	1.09	0.00	3.25	**2.33**	0.92	0.00	1.23
r40x5a-1	928.10	928.10	9.32	1.14	0.00	305.25	8.95	1.32	0.00	140.31	**3.30**	3.99	0.00	14.67
r40x5a-2	888.40	888.40	8.86	0.94	0.00	98.34	8.83	1.04	0.00	86.31	**2.80**	2.25	0.00	11.88
r40x5a-3	947.30	947.30	7.66	2.34	0.00	158.27	7.47	2.48	0.00	76.63	**3.02**	3.09	0.00	11.36
r40x5b-1	1052.00	1052.00	10.60	1.31	0.00	3694.45	10.26	2.80	0.00	3115.92	**5.78**	5.49	0.00	10.49
r40x5b-2	981.50	981.50	8.92	1.38	0.00	10.25	8.57	1.26	0.00	7.61	**2.09**	1.63	0.00	3.77
r40x5b-3	964.30	964.30	5.21	1.48	0.00	11.36	4.51	2.32	0.00	12.33	**2.00**	1.67	0.00	2.68
Average			6.45	1.02	0.00	366.50	6.05	1.35	0.00	288.91	**2.65**	1.87	0.00	5.71

Table 4.VIII: Comparison with the methods of Belenguer et al. [19] and Contardo et al. [36] on family $\mathscr{F}_3$

Instance	z_{UB}	z^*	BMW							CCG-BCP						
			gap_1	t_1	gap_2	t_2	gap_3	t_3	t	gap_1	t_1	gap_2	t_2	gap_3	t_3	t
Perl-12x2	203.98	203.98	1.50	0.30	0.00	0.20	0.00	0.00	0.50	**0.00**	0.02	0.00	0.01	0.00	0.00	0.03
Gas-21x5	424.90	424.90	2.40	3.10	0.00	0.80	0.00	0.00	3.90	**1.61**	0.25	0.00	0.08	0.00	0.00	0.33
Gas-22x5	585.11	585.11	1.50	5.40	0.00	0.60	0.00	0.00	6.00	**0.10**	0.05	0.00	0.18	0.00	0.00	0.23
Min-27x5	3062.02	3062.02	3.00	39.10	0.00	7.90	0.00	0.00	47.00	**0.00**	0.21	0.00	0.10	0.00	0.00	0.31
Gas-29x5	512.10	512.10	7.20	110.70	0.00	67.50	0.00	0.00	178.20	**1.88**	0.44	0.00	3.76	0.00	0.00	4.20
Gas-32x5	562.22	562.22	6.00	13.00	0.10	45.60	0.00	4.80	63.40	**1.24**	0.57	**0.00**	5.95	0.00	0.00	6.52
Gas-32x5b	504.33	504.33	2.60	99.60	0.00	18.30	0.00	0.00	117.90	**0.01**	0.51	0.00	0.19	0.00	0.00	0.70
Gas-36x5	460.37	460.37	5.50	1.60	0.00	1.30	0.00	0.00	2.90	**0.00**	1.04	0.00	0.00	0.00	0.00	1.04
Chr-50x5ba	565.62	565.62	5.80	48.90	0.00	44.50	0.00	0.50	93.90	**1.58**	6.53	0.00	5.55	0.00	0.00	12.08
Chr-50x5be	565.60	565.60	6.00	47.10	0.00	65.80	0.00	0.00	112.90	**2.14**	8.70	0.00	18.66	0.00	0.00	27.36
Perl-55x15	1112.06	1112.06	3.10	102.20	0.00	189.00	0.00	0.00	291.20	**1.96**	197.21	0.00	43.54	0.00	0.00	240.75
Chr-75x10ba	886.30	**844.40**								**7.27**	1243.61	**0.48**	5749.77	**0.00**	13.04	7006.42
Chr-75x10be	848.85	848.85	7.80	1330.40	0.10	2072.60	0.00	10.50	3413.50	**6.71**	485.76	**0.00**	3945.23	0.00	0.00	4430.99
Chr-75x10bmw	802.08	802.08	7.80	1004.70	0.50	790.30	0.00	1031.90	2826.90	**6.02**	207.56	**0.00**	1733.37	0.00	0.00	1940.93
Perl-85x7	1622.50	1622.50	2.80	221.90	0.00	266.20	0.00	0.00	488.10	**1.65**	76.12	0.00	101.80	0.00	0.00	177.92
Chr-100x10	833.43	833.43	6.80	2609.90	0.30	9898.60	0.00	566.20	13074.70	**1.81**	419.35	**0.00**	1130.42	0.00	0.00	1549.77
Average BMW[1]			4.65	375.86	0.07	897.95	0.00	107.59	1381.40	**2.12**	165.50	**0.03**	796.16	0.00	0.81	962.47

[1] Average on instances reported by Baldacci et al. [16].

Table 4.IX: Comparison with the method of Baldacci et al. [16] on family $\mathscr{F}_1$

Instance	z_{UB}	z^*	BMW							CCG-BCP						
			gap_1	t_1	gap_2	t_2	gap_3	t_3	t	gap_1	t_1	gap_2	t_2	gap_3	t_3	t
ppw-20x5-0a	54793	54793	3.60	8.00	0.10	2.10	0.00	0.10	10.20	**3.06**	0.32	**0.00**	0.32	0.00	0.00	0.64
ppw-20x5-0b	39104	39104	2.10	8.70	0.00	9.20	0.00	0.00	17.90	**0.00**	0.03	0.00	0.00	0.00	0.00	0.03
ppw-20x5-2a	48908	48908	**0.80**	1.60	0.00	2.20	0.00	0.00	3.80	2.40	0.16	0.00	0.33	0.00	0.00	0.49
ppw-20x5-2b	37542	37542	3.70	12.10	0.00	32.70	0.00	0.00	44.80	**0.00**	0.02	0.00	0.00	0.00	0.00	0.02
ppw-50x5-0a	90111	90111	**5.90**	45.80	0.30	6.70	0.00	0.40	52.90	5.96	20.50	**0.11**	12.88	0.00	0.10	33.48
ppw-50x5-0b	63242	63242	5.10	467.10	2.10	92.90	0.00	8368.90	8928.90	**4.01**	9.00	**0.58**	648.79	0.00	503.83	1161.62
ppw-50x5-2a	88298	88298	6.10	8.40	1.40	10.50	0.00	52.70	71.60	**4.05**	11.39	**0.07**	6.31	0.00	0.05	17.75
ppw-50x5-2b	67340	**67308**	6.10	69.00	2.70	75.90	2.70	9386.90	9531.80	**3.77**	3.70	**0.00**	227.31	**0.00**	0.00	231.01
ppw-50x5-2a	84055	84055	4.00	7.80	0.60	20.50	0.00	30.20	58.50	**2.01**	4.48	**0.01**	27.30	0.00	0.08	31.86
ppw-50x5-2b	51822	51822	6.50	55.90	0.00	136.10	0.00	0.00	192.00	**0.69**	1.99	0.00	9.79	0.00	0.00	11.78
ppw-50x5-3a	86203	86203	6.10	20.20	1.00	18.80	0.00	22.50	61.50	**3.92**	33.37	**0.82**	30.94	0.00	107.65	171.96
ppw-50x5-3b	61830	61830	5.50	45.00	0.30	80.40	0.00	5.80	131.20	**2.53**	8.29	**0.00**	74.80	0.00	0.00	83.09
ppw-100x5-0a	274814	274814	**1.20**	292.30	0.20	63.70	0.00	46.60	402.60	4.55	514.97	0.20	322.41	0.00	133.69	971.07
ppw-100x5-0b	214392	**213568**	**0.72**	773.60	**0.42**	91.00	0.42	8869.60	9734.20	3.14	391.58	0.43	6858.13	0.29	12354.60	19604.40
ppw-100x5-2a	193671	193671	**1.30**	91.10	0.10	23.10	0.00	2.30	116.50	3.52	112.82	**0.06**	22.22	0.00	0.66	135.70
ppw-100x5-2b	157173	**157095**	**1.80**	2419.30	0.40	624.20	0.40	12415.40	15458.90	2.14	146.75	**0.10**	15324.80	**0.00**	5392.35	20863.90
ppw-100x5-3a	200079	200079	**2.10**	227.00	0.20	31.70	0.00	14.70	273.40	3.59	118.87	**0.17**	67.38	0.00	38.95	225.20
ppw-100x5-3b	152441	152441	**2.00**	734.20	0.10	270.50	0.00	14.80	1019.50	2.08	70.44	**0.00**	539.20	0.00	0.00	609.64
ppw-100x10-0a	289017	289017	**2.70**	257.50	1.90	115.60	1.90	23089.40	23462.50	5.76	4411.79	**1.69**	1627.38	**1.35**	42053.90	48093.10
ppw-100x10-0b	234641	234641	**3.20**	426.60	2.20	437.60	2.20	19278.00	20142.20	4.39	581.21	**2.08**	20735.90	**2.04**	22541.80	43858.80
ppw-100x10-2a	243590	243590	**2.60**	275.70	0.50	65.50	0.00	7495.60	7836.80	2.82	589.95	**0.39**	725.90	0.00	5677.72	6993.57
ppw-100x10-2b	203988	203988	1.90	842.20	0.10	882.20	0.00	31.50	1755.90	**0.80**	210.36	**0.00**	339.55	0.00	0.00	549.91
ppw-100x10-3a	252421	252421	6.20	100.50	2.10	137.00	2.10	14558.70	14796.20	**6.02**	4011.62	**1.91**	1231.36	**1.53**	33510.20	38753.20
ppw-100x10-3b	204597	204597	4.40	504.10	1.60	529.70	1.60	19289.50	20323.30	**4.00**	616.12	**1.55**	15007.20	**1.53**	23181.40	38804.70
ppw-200x10-1a	479425	478845								**8.55**	12970.70	**1.15**	12418.10	**1.04**	32031.80	57420.70
ppw-200x10-1b	378773	378773								**5.32**	11168.90	**1.18**	383307.00	**1.18**	157438.00	551914.00
ppw-200x10-2a	450468	450468								**5.06**	1107.24	**0.87**	9552.10	**0.82**	21913.90	32573.20
ppw-200x10-2b	374435	374435								**3.04**	1509.01	**0.40**	68446.80	**0.36**	76737.80	146694.00
ppw-200x10-3a	472898	**469433**								**6.61**	13541.20	**0.13**	11677.20	**0.00**	11579.70	36798.00
ppw-200x10-3b	364178	364178								**4.92**	9072.59	**1.01**	56865.70	**1.01**	15068.20	81006.50
Average BMW[1]			3.57	320.57	0.76	156.66	0.47	5123.90	5601.13	**3.13**	494.57	**0.42**	2660.01	**0.28**	6062.37	9216.95
Average on solved by BMW			3.56	184.89	0.41	102.87	0.00	946.24	1234.00	**2.71**	100.41	**0.14**	166.36	**0.00**	380.16	646.93
Average on solved by CCG-BCP			3.60	296.39	0.53	128.89	0.16	1994.13	2419.41	**2.73**	97.76	**0.13**	967.38	**0.00**	623.95	1689.09

[1] Average on instances reported by Baldacci et al. [16].

Table 4.X: Comparison with the method of Baldacci et al. [16] on family $\mathscr{F}_2$

Instance	z_{UB}	z^*	BMW							CCG-BCP						
			gap_1	t_1	gap_2	t_2	gap_3	t_3	t	gap_1	t_1	gap_2	t_2	gap_3	t_3	t
r30x5a-1	819.5	819.5	3.30	49.20	0.70	24.80	0.00	1.40	75.40	**2.89**	0.60	**0.08**	1.85	0.00	0.02	2.47
r30x5a-2	821.5	821.5	5.40	88.60	1.60	27.50	0.00	5.80	121.90	**3.73**	0.38	**0.00**	3.34	0.00	0.00	3.72
r30x5a-3	702.3	702.3	3.70	35.80	0.00	30.50	0.00	0.00	66.30	**0.00**	0.44	0.00	0.06	0.00	0.00	0.50
r30x5b-1	880.0	880.0	6.40	75.40	0.00	21.50	0.00	0.00	96.90	**2.69**	1.01	0.00	3.56	0.00	0.00	4.57
r30x5b-2	825.3	825.3	3.00	50.30	0.00	6.80	0.00	0.00	57.10	**1.22**	0.97	0.00	0.27	0.00	0.00	1.24
r30x5b-3	884.6	884.6	**1.30**	31.90	0.00	7.30	0.00	0.00	39.20	2.33	0.92	0.00	0.31	0.00	0.00	1.23
r40x5a-1	928.1	928.1	6.60	169.60	0.00	99.50	0.00	0.00	269.10	**3.30**	3.99	0.00	10.68	0.00	0.00	14.67
r40x5a-2	888.4	888.4	5.60	181.20	0.20	78.80	0.00	0.70	260.70	**2.80**	2.25	**0.19**	9.58	0.00	0.05	11.88
r40x5a-3	947.3	947.3	4.90	158.80	0.10	84.50	0.00	0.80	244.10	**3.02**	3.09	**0.00**	8.27	0.00	0.00	11.36
r40x5b-1	1052.0	1052.0	**5.50**	159.70	0.00	70.60	0.00	0.00	230.30	5.78	5.49	0.00	5.00	0.00	0.00	10.49
r40x5b-2	981.5	981.5	6.20	213.90	0.00	82.90	0.00	0.00	296.80	**2.09**	1.63	0.00	2.14	0.00	0.00	3.77
r40x5b-3	964.3	964.3	3.30	209.70	0.00	21.50	0.00	0.00	231.20	**2.00**	1.67	0.00	1.01	0.00	0.00	2.68
Average			4.60	118.67	0.22	46.35	0.00	0.72	165.75	**2.65**	1.87	**0.02**	3.84	0.00	0.01	5.71

Table 4.XI: Comparison with the method of Baldacci et al. [16] on family $\mathscr{F}_3$

Instance	z_{UB}	z^*	BMW							CCG-BCP						
			gap_1	t_1	gap_2	t_2	gap_3	t_3	t	gap_1	t_1	gap_2	t_2	gap_3	t_3	t
P111112	1467.68	1467.68	8.70	1471.60	0.20	3039.80	0.00	57.60	4569.00	**4.63**	885.82	**0.00**	3581.25	0.00	0.00	4467.07
P111212	1394.80	1394.80	9.00	1571.30	0.40	5122.90	0.00	416.30	7110.50	**4.67**	859.13	**0.00**	20689.10	0.00	0.00	21548.23
P112112	1167.16	1167.16	4.70	1518.40	0.00	2503.50	0.00	0.00	4021.90	**2.74**	75.47	0.00	235.30	0.00	0.00	310.77
P112212	791.66	791.66	6.00	1818.90	0.10	4100.60	0.00	15.50	5935.00	**1.36**	21.32	**0.00**	1818.15	0.00	0.00	1839.47
P113112	1245.45	**1238.24**	9.10	2615.30	1.60	17579.50	1.60	37716.50	57911.30	**3.95**	454.58	**0.61**	118970.00	**0.34**	52278.40	171702.98
P113212	902.26	902.26	5.20	2755.00	0.00	4509.60	0.00	0.00	7264.60	**0.49**	51.30	0.00	101.93	0.00	0.00	153.23
P131112	1900.70	**1892.17**	7.50	2408.70	1.00	7156.60	1.00	27217.40	36782.70	**6.24**	11926.90	**0.85**	299823.00	**0.64**	38047.20	349797.10
P131212	1965.12	1965.12	**8.00**	2165.30	1.00	6686.30	1.00	17537.90	26389.50	8.43	10682.50	**0.95**	207942.00	**0.82**	33676.30	252300.80
P132112	1443.33	1443.32	**3.00**	19150.40	0.00	19081.70	0.00	70.90	38303.00	5.68	3582.12	0.00	15021.90	0.00	0.00	18604.02
Average			6.80	3941.66	0.48	7753.39	0.40	9225.79	20920.83	**4.24**	3171.02	**0.27**	74242.51	**0.20**	13777.99	91191.52
Average on solved instances			6.10	4714.27	0.12	6393.02	0.00	93.38	11200.67	**3.26**	912.53	**0.00**	6907.94	0.00	0.00	7820.47

Table 4.XII: Comparison with the method of Baldacci et al. [16] on family $\mathscr{F}_4$

Instance	z_{UB}	z^*	BMW							CCG-BCP						
			gap_1	t_1	gap_2	t_2	gap_3	t_3	t	gap_1	t_1	gap_2	t_2	gap_3	t_3	t
M-n150x14a	1352.93	1352.93	**7.60**	1266.20	0.20	89144.50	0.00	5320.10	95730.80	9.61	34342.60	**0.09**	351540.00	0.00	1.32	385883.92
M-n150x14b	1212.46	1212.46	**7.30**	2499.80	0.40	48694.00	0.00	324.00	51517.80	7.50	39786.80	**0.20**	352454.00	0.00	30.68	392271.48
M-n199x14a	1644.35	1644.35	**6.50**	14428.10	0.30	188049.90	0.00	606.50	203084.50	12.06	64412.40	**0.15**	616522.00	0.00	10.81	680945.21
M-n199x14b	1480.43	1480.43	**7.40**	6187.60	0.10	259498.30	0.00	149.60	265835.50	10.22	67992.50	**0.09**	1006620.00	0.00	1.98	1074614.48
Average			**7.20**	6095.43	0.25	146346.67	0.00	1600.05	154042.15	9.85	51633.57	**0.13**	581784.00	0.00	11.20	633428.77

Table 4.XIII: Comparison with the method of Baldacci et al. [16] on family $\mathscr{F}_5$

CHAPTER 5

A GRASP + ILP-BASED METAHEURISTIC

Notes about the chapter

The contents of this chapter correspond to those of the article entitled *A GRASP + ILP-based Metaheuristic for the Capacitated Location-Routing Problem*, co-authored with Professors Jean-François Cordeau and Bernard Gendron, which is going to be submitted for publication to Journal of Heuristics (ISSN: 1572-9397). Preliminary results have also been presented in the *Optimization Days 2011 Conference*, in Montréal, Canada (2011) and in the *CORS Annual Conference 2011*, in St-Johns, Canada (2011).

A GRASP + ILP-based Metaheuristic for the Capacitated Location-Routing Problem

Claudio Contardo[1,3], Jean-François Cordeau[2,3], Bernard Gendron[1,3]

[1]Département d'informatique et de recherche opérationnelle, Université de Montréal
C.P. 6128, succ. Centre-ville, Montréal (PQ) Canada H3C 3J7

[2]Canada Research Chair in Logistics and Transportation and HEC Montréal
3000 chemin de la Côte-Sainte-Catherine, Montréal (PQ) Canada H3T 2A7

[3]Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT)
C.P. 6128, succ. Centre-ville, Montréal (PQ) Canada H3C 3J7

In this paper we present a three-phase heuristic method for the Capacitated Location-Routing Problem. In the first stage, we apply a GRASP followed by local search procedures to construct a bundle of solutions. In the second stage, an integer-linear program (ILP) is solved taking as input the different routes belonging to the solutions of the bundle, with the objective of constructing a new solution as a combination of these routes. In the third and final stage, the same ILP is iteratively solved by column generation to improve the solutions found during the first two stages. The last two stages are based on a new model introduced in this paper, the location-reallocation model, which generalizes the capacitated facility location problem and the reallocation model by simultaneously locating facilities and reallocating customers to routes assigned to these facilities. Extensive computational experience shows that our method is competitive with the methods found in the literature, yielding the tightest average gaps on several sets of instances and being able to improve the best known feasible solutions for some of them.

Key words: location-routing, column generation, metaheuristic.

5.1 Introduction

In the capacitated location-routing problem (CLRP) we are given a set of potential facilities I and a set of customers J. To each facility $i \in I$ we associate a fixed setup cost

f_i and a capacity b_i. To each customer $j \in J$ we associate a demand d_j. An unlimited, homogeneous fleet must be routed from the open facilities to serve the demand of the customers in J. To each vehicle is associated a capacity Q, and to every two nodes i and j is associated a traveling cost c_{ij}. The goal is to select a subset of facilities and to design vehicle routes around these facilities in order to 1) visit each customer once, 2) respect both vehicle and facility capacities and 3) minimize the total cost.

The CLRP is an $\mathscr{NP}$-hard combinatorial optimization problem since it generalizes two well known $\mathscr{NP}$-hard problems: the capacitated facility location problem (CFLP) and the capacitated vehicle routing problem (CVRP). Exact methods for this problem include branch-and-cut methods [19, 36] and column generation methods [16, 37]. These methods are able to solve instances with up to 200 customers. However, some instances with 100 customers still remain unsolved. To handle large size instances, Prins et al. [120, 122], Prodhon [124, 125] propose several metaheuristics. The method based on Lagrangean relaxation with cooperative granular tabu-search seems to be the most effective for handling large instances of the CLRP. This method combines the solution of an integer-linear program (ILP) (a CFLP) solved by Lagrangean relaxation (for location decisions) followed by a granular tabu-search (for routing decisions). Pirkwieser and Raidl [116] have introduced a variable neighborhood search (VNS) algorithm for the periodic CLRP (PLRP) and the CLRP based on the combination of a pure VNS along with the solution of several ILPs. The ILPs they consider include a location model (a two-index CFLP) and a reallocation model (a set partitioning model). Hemmelmayr et al. [78] have developed an adaptive large neighborhood search (ALNS) heuristic for the CLRP. In an ALNS method, several different neighborhoods are applied and ranked on-the-run according to their success to improve solutions. In the subsequent iterations the highest ranked neighborhoods have a larger probability of being chosen. Their algorithm is capable of improving the best known solutions on several instances. Finally, Yu et al. [147] propose a simulated annealing heuristic for the problem, in which CLRP solutions are coded as genes and then modified using mutation and crossover operators.

The main contributions of this paper are:

i. to introduce a new greedy randomized adaptive search procedure (GRASP) for the

CLRP that is competitive with the previous GRASP proposed by Prins et al. [121] and which provides better average gaps on several sets of instances.

ii. to introduce a novel location-reallocation model that takes into account the location and the routing decisions simultaneously. The proposed model is based on a set-partitioning formulation that generalizes both the CFLP and the reallocation model of de Franceschi et al. [48], the first by adding the possibility of inserting customers in the middle of the routes, and the second by adding the possibility of reallocating whole routes to different facilities.

iii. to introduce a new technique based on the solution of an ILP, for combining a bundle of reasonably good solutions with the objective of eventually producing another solution of better quality.

The location-reallocation model introduced here can also be seen as a restricted CLRP in which some routing decisions are fixed, and thus also inherits all of the cuts valid for the CLRP [19, 36]. The addition of these extra cuts plays an important role in the proposed heuristic. Indeed, the strength of the model relies on the quality of the root relaxation lower bound. As a pure branch-and-cut-and-price algorithm is computationally too demanding, column generation is applied only at the root node, and even there by only applying some simple pricing heuristics. The resulting ILP is then solved by means of a general-purpose solver. Therefore, the strength of the linear relaxation lower bound is crucial for the performance of the algorithm.

The rest of the paper is organized as follows: In Section 5.2 we give a general description of our solution approach. In Section 5.3 we present two of the metaheuristics that are used in our algorithm, namely a GRASP and a local search procedure used to improve solutions. In Section 5.4 we introduce the location-reallocation model (LRM). We strengthen it with valid inequalities and describe the pricing algorithm used to derive columns of negative reduced cost. In Section 5.5 we introduce the two hybrid metaheuristics, namely a solution blender heuristic and a local improvement heuristic, both of which are based on the solution of the LRM. This is followed by computational results in Section 5.6 and by conclusions in Section 5.7.

5.2 An overview of the complete algorithm

In this section we give a general description of the different parts of our algorithm, and describe it by means of a pseudo-code. Our algorithm consists of four main procedures, namely a GRASP metaheuristic, local search (LS), a solution blender (SB) and a local improvement heuristic (LIH).

5.2.1 GRASP

A GRASP is a simple metaheuristic based on the randomization of a greedy criterion. In this paper, we propose a GRASP based on a variation of the extended Clarke and Wright savings algorithm (ECWSA) introduced by Prins et al. [121].

5.2.2 Local search

Local search procedures are greedy algorithms applied to a feasible solution to further improve its quality. Here, we use seven different methods that are applied iteratively until no further improvements are found.

5.2.3 Solution blender

The solution blender (SB) is a method based on the solution of an integer-linear program, called the location-reallocation model (LRM). The LRM is a set-partitioning model in which three types of variables are considered: location variables, assignment variables and routing variables. The first two are polynomial in number while there are an exponential number of the latter. Normally, such models are solved by column generation. However, in the SB the set of routing variables is restricted to contain a fixed number of columns defined in advance, and therefore no column generation is applied. We complement this with the use of local branching constraints used to fix a large number of variables.

5.2.4 Local improvement heuristic

The local improvement heuristic (LIH) is a *destroy-and-repair* method inspired from the ALNS metaheuristic. In this method, a destroy operator is applied to remove customers from the current solution. The LRM is then solved by column generation, with the aim of constructing a new feasible solution of better quality. The LIH uses a parameter $\Gamma \leq |J|$ in the destroy operators to remove a target number Γ of customers from the solution which we denote it by $LIH(\Gamma)$.

5.2.5 The complete algorithm

We now describe by means of a pseudo-code the complete algorithm. For a given solution $\mathscr{T}$ of the CLRP, let $v(\mathscr{T})$ denote the cost of $\mathscr{T}$. Also, let Γ_0 be a parameter representing a certain number of customers, normally a small proportion of them.

Algorithm 5.1 GRASP + ILP

1: Use GRASP + LS and build solution pool $\mathscr{P}$.
2: Use SB and add the new solutions found to $\mathscr{P}$.
3: $\mathscr{T} \leftarrow arg\,min\{v(\mathscr{S}) : \mathscr{S} \in \mathscr{P}\}$.
4: $\Gamma \leftarrow \Gamma_0$.
5: **repeat**
6: Apply $LIH(\Gamma)$ to $\mathscr{T}$.
7: **if** it found a solution $\mathscr{T}' \notin \mathscr{P}$ **then**
8: $\mathscr{P} \leftarrow \mathscr{P} \cup \mathscr{T}'$.
9: **if** $v(\mathscr{T}') < v(\mathscr{T})$ **then**
10: $\mathscr{T} \leftarrow \mathscr{T}'$ and **go to** 6.
11: **end if**
12: **end if**
13: Use SB and add the new found solutions to $\mathscr{P}$.
14: **if** a new solution $\mathscr{T}'$ was found with $v(\mathscr{T}') < v(\mathscr{T})$ **then**
15: $\mathscr{T} \leftarrow \mathscr{T}'$ and **go to** 6.
16: **end if**
17: Increase Γ by some positive value.
18: **until** some stopping criterion is met

5.3 Pure metaheuristics

In this section we describe two metaheuristic procedures used in our algorithm, namely a GRASP and a local search (LS) method. We refer to these as pure metaheuristics to distinguish them from the ILP-based metaheuristics that will be introduced later.

5.3.1 GRASP

GRASP is a popular metaheuristic which, based on some simple greedy deterministic criterion, includes some randomization in order to diversify the search of the solution

space. This randomized greedy algorithm is applied many times, thus increasing the likelihood of identifying a good quality solution. The randomization is usually subject to what is called a restricted candidate list (RCL), for which a given greedy criterion of the form *"pick* $x' = arg\,min_x\{f(x) : x \in X\}$*"* is replaced with *"Let* $\mathscr{L}$ *contain the* κ *elements* $x \in \mathscr{X}$ *with smallest value of* $f(x)$*. Pick* x' *randomly in* $\mathscr{L}$*"*. For the CLRP, Prins et al. [121] proposed a GRASP method that they complemented with path relinking. Their method is based on the so-called extended Clarke and Wright savings algorithm (ECWSA). In this paper we propose a variant to that method, and explain how we apply randomization at three different levels of the algorithm. We now describe, by means of a pseudo-code (Algorithm 5.2), the deterministic algorithm on which we base the proposed GRASP algorithm.

First, let us introduce some notation. For any two routes R, S and for any facility $i \in I$, $s(R,S,i)$ represents the saving produced when merging routes R and S for creating a new route T which is assigned to facility i, and such that capacities are respected. Note that if R and S contain two or more customers, four different merges are possible, and so the definition of s implicitly assumes that the resulting route T is the one with the lowest cost. For details on the merging procedure, the reader is referred to Clarke and Wright [35] and to Prins et al. [121]. Also, for a boolean statement p, we define δ_p to be equal to 1 if $p = true$, and 0 otherwise. Finally, F denotes the set of currently open facilities, A denotes the set of already assigned customers, $\gamma(\cdot)$ represent the facilities to which customers are assigned (a customer $j \notin A$ is such that $\gamma(j) = -1$), and $l(\cdot)$ represents the current loads of facilities.

Algorithm 5.2 ECWSA

1: $F \leftarrow \emptyset, A \leftarrow \emptyset, \gamma(j) \leftarrow -1$ for all $j \in J$, $l(i) \leftarrow 0$ for all $i \in I$.
2: **while** $\exists j \in J, \gamma(j) = -1$ **do**
3: $j' \leftarrow arg\,min\{\sum_{i \in F} c_{ij} : j \notin A\}$.
4: $i' \leftarrow arg\,min\{2c_{ij'} + f_i \delta_{i \notin F} : i \in I, l(i) + d_{j'} \leq b_i\}$.
5: $F \leftarrow F \cup \{i'\}, A \leftarrow A \cup \{j'\}, \gamma(j') \leftarrow i', l(i') \leftarrow l(i') + d_{j'}$.
6: **end while**
7: $\mathscr{R} \leftarrow \{\{\gamma(j), j\} : j \in J\}$.
8: **repeat**
9: $(R', S', i') \leftarrow arg\,max\{s(R, S, i) : R, S \in \mathscr{R}, i \in I,$ and merge respects capacities$\}$.
10: $s \leftarrow s(R', S', i')$.
11: **if** $s > 0$ **then**
12: Merge R', S' into a new route T' and assign ir to facility i'.
13: Update $\mathscr{R}$ by replacing R' and S' by the merged route T'.
14: Update F, A, γ and l accordingly.
15: **end if**
16: **until** $s \leq 0$

In our GRASP, we replace the three optimization problems appearing in the pseudo-code with some randomized versions. The deterministic statement $j' \leftarrow arg\,min\{\sum_{i \in F} c_{ij} : j \notin A\}$ is changed to randomly picking a customer j' among the five customers not in A with minimum value of $\sum_{i \in F} c_{ij}$. The statement $i' \leftarrow arg\,min\{2c_{ij'} + f_i \delta_{i \notin F} : i \in I, l(i) + d_{j'} \leq b_i\}$ is decomposed into two random stages. For the set of closed facilities (if any), we compute the quantity $v(F^c) = (\sum_{i \notin F} 2c_{ij'} + f_i)/|F^c|$ and assign to this quantity a dummy node i_{F^c}, and for each facility $i \in F$ we compute separately the quantity $v(i) = 2c_{ij'}$ and assign to it the node i. Now, we put in a list the $|F| + 1$ quantities defined before (only $|F|$ in case $|F^c| = 0$) and randomly pick a node i' among the three which minimize it. If $i' \in I$, then we assign customer j' to this facility. Otherwise, if $i' = i_{F^c}$ we randomly pick a facility $i'' \notin F$ among the $k = \lceil |I|/3 \rceil$ that minimize $2c_{i''j'} + f_{i''}$. Facility i'' is then opened and customer j' assigned to it. Finally, the state-

ment $(R', S', i') \leftarrow arg\,max\{s(R,S,i) : R,S \in \mathscr{R}, i \in I, \text{and merge respects capacities}\}$ is modified to randomly pick a merge among the five possibe merges with maximum saving. We call this algorithm the randomized ECWSA (RECWSA). The RECWSA is repeated for 300 times, and the solutions are stored in a solution pool $\mathscr{P}$. For each of the solutions in the pool, we apply local search (detailed in the next section) to improve their quality. After that, we clean the pool by keeping the 100 best solutions. These solutions will be the input of the solution blender heuristic that will be described in Section 5.5.1.

5.3.2 Local Search

Local search procedures are simple greedy algorithms applied to a feasible solution to further improve its quality. They are usually based on simple greedy criteria, which are fast to compute. In our case, we have implemented seven different local search procedures:

FACILITY OPEN Compute the cost of opening a previously closed facility i and of re-assigning routes to this newly open facility. We potentially close a facility if it is cheaper to move all of its routes to the newly open one. This procedure is performed using a first-improvement criterion (a move is accepted as soon as it produces another solution of lower cost).

FACILITY SWAP Swap an open facility with a closed one, and reassign routes from one facility to the other. This procedure is performed using a first improvement criterion.

GIANT TOUR SPLIT Merge all the routes linked to the same facility into one giant TSP tour [132]. Split the tour using a shortest path algorithm so as to minimize the total routing cost. This procedure is performed using a first-improvement criterion.

ROUTE SWAP Swap two routes linked to different facilities. This procedure is performed using a first-improvement criterion.

2-OPT Swap two customers from different routes [44]. This procedure is performed using a best-improvement criterion (among the moves producing solutions with lower objective values, the one that produces the lowest cost solution is accepted).

2-OPT* Two routes are split and re-merged [119]. This procedure is performed using a best-improvement criterion.

3-OPT Pick three customers in different routes and evaluate all possible swaps between them [98]. This procedure is performed using a first-improvement criterion.

Each of these procedures is performed repeatedly until no further improvements are detected. Also, the order in which each of the procedures is performed is as described above, and they are cyclically performed until no further improvements are found.

5.4 A location-reallocation model

In this section we introduce the Location-Reallocation Model (LRM), a new ILP model that generalizes the CFLP and the reallocation model of de Franceschi et al. [48], the first by adding the routing decisions into the problem, and the second the location decisions. This model is the core of the ILP-based heuristics introduced in this paper, namely the solution blender and the local improvement heuristics. We present a mathematical formulation of the model, some valid inequalities and the pricing algorithm used in the column generation.

5.4.1 Mathematical formulation

Let us consider a feasible solution $\mathscr{T}$ of the CLRP. For a given customer subset $T \subseteq J$ let $\mathscr{T}(T)$ be the truncated solution of the CLRP obtained from $\mathscr{T}$ after

i. removing the customers of set T,

ii. short-cutting the remaining consecutive nodes in the routes,

iii. deleting the edges linking facilities to customers,

iv. and relinking the two remaining endpoints of every route.

As a result, what we obtain is a set of closed subtours, each of which consisting of at least two customers. Figure 5.1 illustrates this procedure. On the left side, circular dots represent customer locations, whereas square nodes represent facility locations. The nodes surrounded by dotted circles are the nodes in set T. The right side represents

the subtours resulting from the removal of the customers in set T. Let us denote by $\mathscr{R}$ the set of these subtours and for each $r \in \mathscr{R}$ and $i \in I$ let $h(i,r)$ and $t(i,r)$ be the two consecutive nodes in r which, after linking r to i using these two nodes as endpoints, produce the route with the least possible cost. To avoid symmetries, we arbitrarily take $h(i,r), t(i,r)$ satisfying $h(i,r) < t(i,r)$. Customers in T must be reinserted back into $\mathscr{T}(T)$ and subtours $R \in \mathscr{R}$ must be assigned to facilities to construct a (eventually new) feasible solution of the CLRP. For every subtour r we let $E(r), V(r)$ be the sets of edges and customers in that subtour. We also let $c(r)$ be the routing cost of such subtour, and $q(r)$ be its load. For every $i \in I$ and $e \in \cup_{r \in \mathscr{R}} E(r)$ we associate an insertion point $p = (i,e)$, at which customers in T can be reinserted. Let us denote, for a given facility i, $\mathscr{I}_i(\mathscr{R}) = \{p = (i,e) : e \in E(r) \text{ for some } r \in \mathscr{R}\}$. Also, for each $r \in \mathscr{R}$ and for each $i \in I$, $p = (i, \{i, h(i,r)\})$ represents an insertion point from which a subtour can be connected to facility i. For a given $i \in I$, we denote $\mathscr{I}(h(i,\mathscr{R})) = \{p = (i,e) : r \in \mathscr{R}, e = \{i, h(i,r)\}\}$. Analogously, $p = (i, \{i, t(i,r)\})$ represents the other insertion point from which the subtour is linked to facility i and we denote the set of insertion points as $\mathscr{I}(t(i,\mathscr{R}))$. Finally, the insertion point $p = (i, \{i,i\})$ is used for routes starting and ending at facility i and serving only customers in T. For every facility $i \in I$ the set of insertion points associated with i is defined as

$$\mathscr{I}_i = \mathscr{I}_i(\mathscr{R}) \cup \mathscr{I}(h(i,\mathscr{R})) \cup \mathscr{I}(t(i,\mathscr{R})) \cup \{(i, \{i,i\})\}. \tag{5.1}$$

For every insertion point $p = (i,e) \in \mathscr{I}_i$ we define $i(p) = i$, $e(p) = e$. Also, note that unless $p = (i, \{i,i\})$, $e(p)$ must contain at least one node in a subtour r, and if both nodes belong to a subtour then it must be the same. Therefore, one can define $r(p)$ equal to r in that case, and equal to -1 in the case $p = (i, \{i,i\})$. For every insertion point p, we denote by $\mathscr{S}_p$ the set of sequences or partial paths that can be inserted in p. Note that all the sequences that result in a violation of the capacities can be safely removed from $\mathscr{S}_p$. For every $s \in \mathscr{S}_p$ we let $E(s)$ be the set of edges defining s, $q(s)$ be the load of s (without considering the two endpoints) and $c(s)$ be the cost associated to that partial

route, computed as follows:

$$c(s) = \begin{cases} \sum_{e \in E(s)} c_e - c_{e(p)} & \text{if } p \in \mathscr{I}_i(\mathscr{R}), s \in \mathscr{S}_p \\ \sum_{e \in E(s)} c_e & \text{otherwise.} \end{cases} \tag{5.2}$$

(a) Complete solution. Set T surrounded by dotted circles

(b) Incomplete solution after the removal of nodes in T

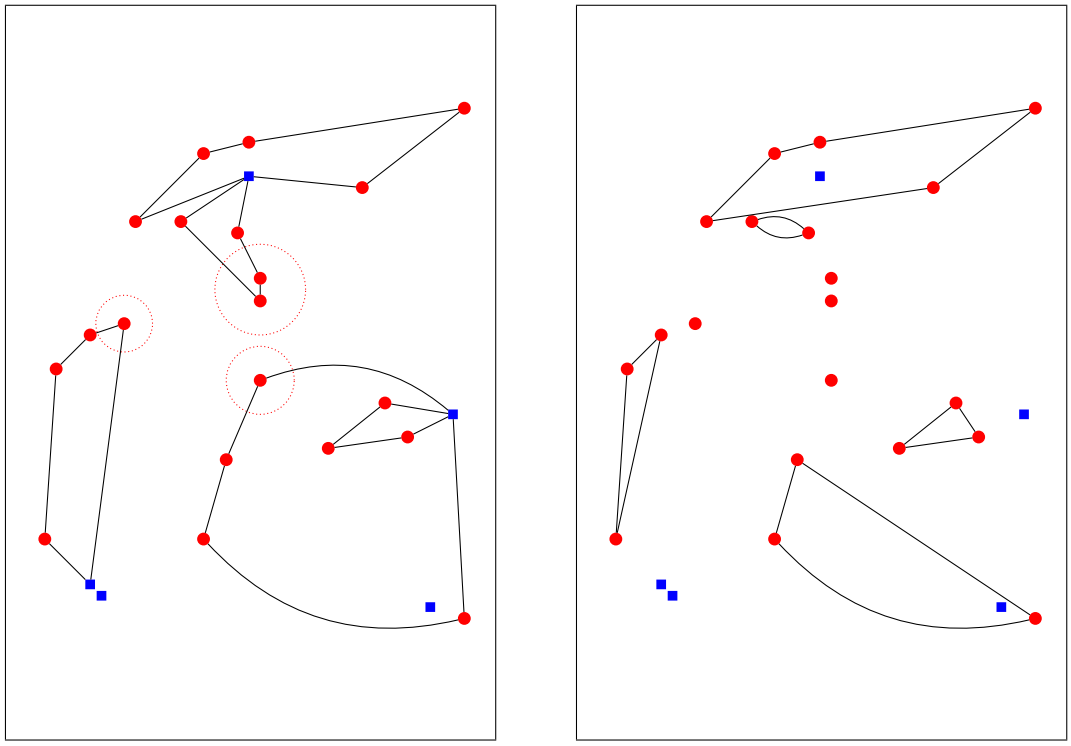

Figure 5.1: Example of node removal from a CLRP solution

Let us define the following notation. Let z_i be a binary variable equal to 1 iff facility i is selected for opening. For every pair $\{i, j\}, i \in I, j \in T$ let y_{ij} be a binary variable equal to 1 iff customer j is served by a single-customer route from facility i. For every subtour $r \in \mathscr{R}$ and for every facility $i \in I$ let $u_{ir}^{\mathscr{R}}$ be a binary variable equal to 1 iff subtour r is assigned to facility i. For every facility $i \in I$ and customer $j \in T$ let u_{ij}^{T} be a binary variable equal to 1 iff customer j is served from facility $i \in I$. For every $s \in \mathscr{S}$ we let w_s be a binary variable equal to 1 iff sequence s (associated to a certain insertion point) is selected. The location-reallocation model is as follows:

$$\sum_{r\in\mathscr{R}} c(r) \quad + \quad \min \quad \sum_{i\in I} f_i z_i - \sum_{i\in I, r\in\mathscr{R}} c_{h(i,r)t(i,r)} u_{ir}^{\mathscr{R}} + 2\sum_{e\in\delta(I)} c_e y_e + \sum_{s\in\mathscr{S}} c(s) w_s \quad (5.3)$$

subject to

$$\sum_{i\in I} u_{ij}^{T} = 1 \qquad j \in T \qquad (5.4)$$

$$\sum_{i\in I} u_{ir}^{\mathscr{R}} = 1 \qquad r \in \mathscr{R} \qquad (5.5)$$

$$y_{ij} + \sum_{p\in\mathscr{I}_i} \sum_{s\in\mathscr{S}_p, j\in V(s)} w_s = u_{ij}^{T} \qquad i \in I, j \in T \qquad (5.6)$$

$$\sum_{s\in\mathscr{S}_{(i,\{i,h(i,r)\})}} w_s = u_{ir}^{\mathscr{R}} \qquad i \in I, r \in \mathscr{R} \qquad (5.7)$$

$$\sum_{s\in\mathscr{S}_{(i,\{i,h(i,r)\})}} w_s - \sum_{s\in\mathscr{S}_{(i,\{i,t(i,r)\})}} w_s = 0 \qquad i \in I, r \in \mathscr{R} \qquad (5.8)$$

$$\sum_{s\in\mathscr{S}_p} w_s \leq u_{ir}^{\mathscr{R}} \qquad i \in I, p \in \mathscr{I}_i(\mathscr{R}) \qquad (5.9)$$

$$\sum_{i\in I} \sum_{p\in\mathscr{I}_i, r(p)=r} \sum_{s\in\mathscr{S}_p} q(s) w_s \leq Q - q(r) \qquad r \in \mathscr{R} \qquad (5.10)$$

$$\sum_{j\in T} d_j u_{ij}^{T} + \sum_{r\in\mathscr{R}} q(r) u_{ir}^{\mathscr{R}} \leq b_i z_i \qquad i \in I \qquad (5.11)$$

$$z, y, u, w \text{ binary} \qquad (5.12)$$

The objective function contains two parts: a constant term given by the first expression, which takes into account the cost of the remaining part of the solution after the removal of the nodes in set T; and a linear term, combining setup costs with routing costs. Constraints (5.4)-(5.5) are the assignment constraints of customers to facilities. Constraints (5.6) are the degree constraints which ensure that customers in T will be reinserted. Constraints (5.7)-(5.8) ensure that partial routes $r \in \mathscr{R}$ will be linked to a facility. Constraints (5.9) ensure that for every insertion point $p \in \mathscr{I}_i(\mathscr{R})$ at most one column will be assigned. Moreover, if a route r is not assigned to a certain facility i, then all of the sequences $s \in \mathscr{S}_p$ with $i(p) = i$ and $r(p) = r$ are automatically set to 0. Con-

straints (5.10) are the vehicle capacity inequalities. They make sure that the final routes will not exceed vehicle capacities. Constraints (5.11) are the facility capacity inequalities. They make sure that the total demand assigned to every facility will not exceed its capacity, while at the same time that no load will be assigned to closed facilities.

Note that the minimum sizes of the sequences s may vary. Indeed, a sequence s participates in the construction of multiple-customer routes, so every time we have to make sure that only routes containing two or more customers are generated. Thus, for $p \in \mathscr{I}_i(\mathscr{R})$, the minimum size of $s \in \mathscr{S}_p$ (defined as the number of nodes visited other than those of $e(p)$) is 1. If $p = (i,\{i,i\})$ then the minimum size is 2. Finally, if $p \in \mathscr{I}(h(i,\mathscr{R})) \cup \mathscr{I}(t(i,\mathscr{R}))$ for some i, then the minimum size is 0.

5.4.2 Valid inequalities

The location-reallocation problem described above includes a polynomial number of constraints and can be solved by means of branch-and-price. However, it is possible to include all the valid inequalities from the three-index formulation [37] after the inclusion of the following flow and assignment variables. For every facility $i \in I$ and edge $e \in E$, let us define a flow variable x_e^i as follows:

$$x_e^i = \begin{cases} u_{ir}^{\mathscr{R}} - \sum_{s \in \mathscr{S}_{(i,e)}} w_s & \text{if } e \in E(r) \setminus \{\{h(i,r),t(i,r)\}\} \text{ for some } r \in \mathscr{R} \\ 1 - u_{ir}^{\mathscr{R}} & \text{if } e = \{h(i,r),t(i,r)\} \text{ for some } i \in I, r \in \mathscr{R} \\ \sum_{p \in \mathscr{I}_i} \sum_{s \in \mathscr{S}_p, e \in E(s)} w_s & \text{otherwise.} \end{cases} \tag{5.13}$$

Also, for every facility $i \in I$ and customer $j \in J$ let us define the following assignment variables:

$$u_{ij} = \begin{cases} u_{ir}^{\mathscr{R}} & \text{if } j \in V(r), r \in \mathscr{R} \\ u_{ij}^{T} & \text{if } j \in T. \end{cases} \tag{5.14}$$

Finally, for every facility $i \in I$ and $j \in J \setminus T$ we set $y_{ij} = 0$.

It suffices to use identities (5.13)-(5.14) to include the valid inequalities from the three-index vehicle-flow formulation. In particular, it is useful to include the following

four families of inequalities: y-capacity cuts (y-CC), y-strengthened effective facility capacity inequalities (y-SEFCI), y-location-routing generalized large multistar inequalities (y-LRGLM), and disaggregated co-circuit constraints (DCoCC). For details on the inequalities, we refer to Belenguer et al. [19] and Contardo et al. [36]. Moreover, it is possible to strengthen the y-CC and the y-ESFCI to hybrid forms of the y-strengthened capacity cuts (y-SCC) and set-partitioning strengthened effective facility capacity inequalities (SP-SEFCI), which have been developed by Contardo et al. [36] for solving the CLRP by branch-and-cut-and-price.

5.4.3 Column Generation

The reduced cost of a column w_s will be computed differently depending on the position of its insertion point p. Let $T(s) \subseteq T$ be the set of customers in T that are served by column s. Suppose that no additional inequalities have been added to the problem, and let $\alpha, \beta, \sigma, \gamma, \theta$ be the dual variables associated with constraints (5.6)-(5.10). The reduced cost associated to a column s with an insertion point $p \in \mathscr{I}_i$ will be given by

$$\overline{c}(s) = \begin{cases} c(s) - \sum_{j \in T(s)} \alpha_j - \sum_{j \in T(s)} d_j \theta_{r(p)} - \gamma_p & \text{if } p \in \mathscr{I}_i(\mathscr{R}) \\ c(s) - \sum_{j \in T(s)} \alpha_j - \beta_{ir(p)} - \sigma_{ir(p)} & \text{if } p \in \mathscr{I}(h(i,\mathscr{R})) \\ c(s) - \sum_{j \in T(s)} \alpha_j + \sigma_{ir(p)} & \text{if } p \in \mathscr{I}(t(i,\mathscr{R})) \\ c(s) - \sum_{j \in T(s)} \alpha_j & \text{if } p = (i, \{i,i\}). \end{cases} \tag{5.15}$$

If valid inequalities have been added during the solution of the problem, the reduced costs are modified accordingly using the dual variables associated to these inequalities. Our pricing algorithms take into account the different expressions in (5.15) (modified by the dual information associated to valid inequalities) but they work along the exact same principle. The complete pricing is performed in two stages.

First, we use a simple tabu search heuristic starting from a column containing a single customer. That customer is chosen in such a way that the reduced cost of the resulting column is as small as possible. We use four neigborhoods to inspect the space close to a

given sequence. An ADD neighborhood picks a customer not in the sequence and inserts it into the sequence. A DROP neighborhood is used to perform the opposite move. A SWAP neighborhood picks a customer inside the current sequence and one outside, and swaps them. Finally, a SWITCH neighborhood takes two customers inside the sequence and swaps them. We combine neighborhoods ADD, DROP, SWAP and SWITCH using the customers in set T. The neighborhoods are sorted and applied in the following order: ADD - DROP - ADD - SWAP - ADD - SWITCH. Indeed, preliminary experiments showed that the ADD neighborhhod is often the most useful, and thus it is the one that is performed the most. The movements are using a best-improvement criterion, and we use a tabu list to forbid movements to positions previously visited during the last three iterations. The algorithm stops whenever a column of negative reduced cost has been detected or when a maximum number of iterations has been reached. The maximum number of iterations at the beginning is set to 100. In order to accelerate the pricing algorithms, after seven rounds of adding cuts, we lower this threshold to 20.

When the tabu search procedure finishes with success (i.e., after having identified a column with negative reduced cost), starting from that column we apply a greedy insertion algorithm, very similar to the one presented by de Franceschi et al. [48]. We evaluate the insertion of every single customer in a list $\mathscr{L}$ initially containing the customers in T not yet inserted into the column at every possible position. If the resulting column has negative reduced cost, then it is added to a pool and the same algorithm is recursively applied to it. This dynamic programming algorithm is applied until it reaches a depth of 5 from the starting column (the one obtained by the tabu search procedure).

5.5 ILP-based metaheuristics

In this section we describe two hybrid metaheuristics based on the solution of the LRM described earlier. We first describe a solution blender heuristic (SB), a method based on the existence of a pool of reasonably good solutions. We then describe a local improvement heuristic (LIH) based on the iterative solution of the LRM and solved by column and cut generation.

5.5.1 Solution blender

We present a heuristic procedure based on the solution of a particular case of the LRM. We refer to this method as the solution blender (SB). Given a pool of solutions $\mathscr{P}$, we apply the following procedure to every solution $\mathscr{S} \in \mathscr{P}$. Let $\mathscr{R}(\mathscr{S})$ be the set of routes describing solution $\mathscr{S}$. For every route $R \in \mathscr{R}(\mathscr{S})$ we first consider the subtour produced by disconnecting R from its facility and then reconnecting its two endpoints. This tour is then reconnected to every facility i using as endpoints the pair of consecutive nodes in the subtour that produce the route with minimum cost. This procedure creates, for every route $R \in \mathscr{R}(\mathscr{S})$, $|I|$ routes, each connected to a different facility. We refer to this procedure as the *replication step*.

At the end of the replication step, we will potentially have $\sum_{\mathscr{S} \in \mathscr{P}} |\mathscr{R}(\mathscr{S})| \times |I|$ routes (some repeated routes might be discarded). The LRM is then solved using $T = J$ and by restricting the set of columns to contain those constructed during the replication step, without applying any column generation. The optimal solution of this restricted problem is then likely to combine routes from different solutions. Indeed, in many cases in which the GRASP procedure was not able to find a near optimal solution, the blending phase performed substantially better. In our case, the input for the solution blender is the solution pool $\mathscr{P}$ containing the 100 best solutions found by the GRASP method combined with local search. Every new found solution is also subject to local search. Note also that the blending procedure is a generalization of the procedure introduced by Prins et al. [122] in which the size of the pool is fixed to one. Moreover, in that case this method also coincides with the solution of the CFLP.

5.5.1.1 Local branching

At the end of the root node relaxation, we perform a local branching heuristic to guide the search towards promising directions during the branch-and-bound search. We fix to 1 the location variables whose values are greater than or equal to 0.9. For the location variables that are smaller than or equal to 0.1, we pick at most two variables z_{i_1}, z_{i_2} with

the smallest reduced costs. For these vriables we impose the following constraint:

$$z_{i_1} + z_{i_2} \leq 1.$$

The remaining location variables satisfying $z_i \leq 0.1$ are all fixed to zero. In particular, note that this method gives preference to the variables taking strictly positive values at the root relaxation, over the variables that are at their lower bound 0. In the case where three or more location variables take positive values (all of which having the same reduced cost equal to zero), we give preference to the ones taking the largest values.

5.5.2 Local improvement heuristic

Let $\mathscr{T}$ be the solution with minimum cost resulting from the previous heuristic procedures. Let $\rho = \lceil 0.1|J| \rceil$ be a parameter. For different values of $k > 0$, we let $\Gamma = k\rho$ be the target size of customer set T to be removed from and reinserted back in $\mathscr{T}(T)$. The local improvement phase starts with $\mathscr{T}$ and $k = 1$, and successively solves the LRM using sets T of target size $k\rho$. Each time a better solution is found, the algorithm is restarted with the same value of k. When no more improvement can be detected, k is increased by one unit and the algorithm is restarted. The value of k is increased at most twice, and each time we update this value, we refer to it as a *major iteration* of the local improvement heuristic. Note that every new found solution is subject to local search. In what follows we describe the different parts of this procedure, namely the choice of the customer set T, the inclusion of an initial pool of columns as well as some local branching rules.

5.5.2.1 Choice of set T

The set T of customers to be erased from $\mathscr{T}$ is selected by following similar rules to those explained in de Franceschi et al. [48] and Pirkwieser and Raidl [116]. We first define the following notion of relatedness between two customers: Let $u, v \in J$ be two customer nodes. Let $c_{max} = \max\{c_{hj} : h, j \in J\}$ be the maximum distance between any two customers. We define the relatedness between u and v as $r(u,v) = 1 - c_{uv}/c_{max}$. If

u, v belong to the same route then $r(u,v)$ is multiplied by 0.75, and if u, v belong to the same facility then $r(u,v)$ is multiplied by 0.85. The idea is to penalize the choice of customers belonging to the same route or being served by the same facility, as the local search makes it unlikely that these customers will switch places. The two rules that we have implemented can be summarized as follows:

NEIGHBORHOOD rule Given a pivot customer $\overline{u}$, we make $T = \{\overline{u}\}$ and iteratively insert into T the customer $u \notin T$ such that $\sum_{v \in T} r(u,v)$ is maximal.

RANDOM rule We randomly pick a subset of customers and insert it into T.

We first apply the NEIGHBORHOOD rule five times. Each time, we save into a list N_T the customers that have participated in T in the previous iterations. For the next iteration, we use as pivot node the customer $u \notin N_T$ such that $\sum_{v \neq u, v \notin N_T} r(u,v)$ is maximal. When the NEIGHBORHOOD rule has been used 5 times without success, we use the RANDOM rule five more times.

5.5.2.2 Initial set of columns

We have found it is beneficial to start the column generation algorithm with a small, but likely useful set of initial columns. For every insertion point p, we let $V(p) \subseteq T$ be the subset of customers of size at most five containing the closest nodes to $e(p)$, in terms of the sum of the distances to the two endpoints of $e(p)$. Then, we add to the master problem all the sequences obtained as combinations of the nodes in $V(p)$.

5.5.2.3 Local Branching

Let I^o, I^c the subsets of facilities that are open or closed in solution $\mathscr{T}$. From the beginning of the optimization we let

$$\sum_{i \in I^o} z_i - \sum_{i \in I^c} z_i \geq |I^o| - \eta.$$

Depending on the value of Γ, the parameter η is set either to 2 (if $\Gamma = \rho$) or 0 (if $\Gamma \geq 2\rho$). In the first case, we let at most two location variables change their values, while

in the second case the location variables are actually fixed to their current values in $\mathscr{T}$. When the root node relaxation has been solved with success and no more columns with negative reduced cost or violated inequalities are detected, we also consider the same local branching constraint as for the solution blender.

5.6 Computational experience

We have run our method on an Intel Xeon E5462, 3.0 Ghz processor with 16GB of memory. The code was compiled with the Intel C++ compiler v11.0 and executed on Linux, kernel 2.6. Linear and integer programs were solved with CPLEX 12.2. The algorithm has been tested over four sets of instances from the literature, containing a total of 89 instances. The first set of instances ($\mathscr{F}_1$) has been developed by Belenguer et al. [19] and contains 30 instances with capacitated vehicles and facilities. The second set of instances ($\mathscr{F}_2$) has been introduced by Tuzun and Burke [142] and contains 36 instances with capacitated vehicles and uncapacitated facilities. The third set of instances ($\mathscr{F}_3$) has been adapted from other vehicle routing problems by Barreto [18] and contains 19 instances with capacitated vehicles, mixing some instances with capacitated and uncapacitated facilities. The fourth and last set of instances ($\mathscr{F}_4$) has been introduced by Baldacci et al. [16] and contains four instances with limited vehicle capacities and uncapacitated facilities. The dimensions of the instances vary from very small instances with 12 customers and two facilities up to very large instances with 200 customers and 20 facilities.

For the parameter setting, several runs have been performed on the four sets of instances. At the end, however, we use the same parameters for all instances and the average values reported correspond to those obtained on a total of 10 runs for each instance. In Tables 5.I-5.IV we report the results obtained by our algorithm on all sets of instances. In these tables, z^*_{BKS} corresponds to the best known solution as reported by previous authors, z^*_{avg} is the average cost obtained by our solution method, *stdev* is the standard deviation (in %) of the cost over the 10 runs, gap_{avg} is the average relative gap (in %), computed as $100 \times (z^*_{avg} - z^*_{BKS})/z^*_{BKS}$, T_{avg} is the average CPU time, in sec-

onds, over the 10 runs, z^*_{best} is the best solution found in these 10 runs. This value does not necessarily correspond to the best known solution found by our method during the parameter setting phase, which are described later in Table 5.X. Finally, gap_{best} is the relative gap (in %) of the best solution found, computed as $100 \times (z^*_{best} - z^*_{BKS})/z^*_{BKS}$. As the results show, our solutions are 0.22% above the best known solution on average for the instances of set $\mathscr{F}_1$, 0.59% for the instances of set $\mathscr{F}_2$, 0.61% for the instances of set $\mathscr{F}_3$ and 0.34% for the instances of set $\mathscr{F}_4$. Moreover, we are able to improve these values in 11 out of the 89 instances considered in our study. Regarding the CPU times, they lie around 45 minutes on average, and usually stay below 3 hours.

In Tables 5.V-5.VIII we report the evolution of our algorithm during the different stages. In these tables, instances are grouped according to their size. The headers *GRASP*, *SB*, *LIH 1, 2, 3* stand for the different parts of our algorithm, including the three major iterations of the local improvement heuristic. The sub-headers gap_{avg} and T_{avg} stand for the average relative gap (in percentage, computed as before) and the average CPU time spent in seconds. In general, the SB is a very effective method for reducing the gap with respect to the solutions found during the GRASP. However, the GRASP should not be underestimated, since the behaviour of the SB depends on the good quality of the routes found by the GRASP. For the LIH, it is worth observing that for instances of set $\mathscr{F}_1$ just the first improvement is able to reduce the gap by one half. Subsequent iterations of the improvement stage are able to reduce the gap by smaller margins. Depending on the needs of the decision maker, the improvement phase can be extended to more iterations or reduced to fewer, compensating the time saved or added with the quality of the solutions obtained.

In Table 5.IX we compare our algorithm against several of the most recent heuristics developed for the CLRP. The algorithms considered are: GRASP [121], MA|PM [120], LRGTS [122], VNS+ILP [116], SALRP [147] and ALNS [78]. The set of instances $\mathscr{F}_4$ has not been considered by any of these heuristics and is therefore not included in this table. As shown in the table, our algorithm is able to obtain the tightest average gaps for sets $\mathscr{F}_1$ and $\mathscr{F}_2$, and competitive average gaps on instances of set $\mathscr{F}_3$, getting better average results than GRASP, MA|PM and LRGTS but outperformed by SALRP

and ALNS. On the other hand, algorithms LRGTS and VNS+ILP take much less CPU time, but they seem to be less robust than our method in terms of solution quality. Additionally, our GRASP is able to obtain better solutions than that developed by Prins et al. [121] for instances of families $\mathscr{F}_1$ and $\mathscr{F}_2$. Finally, note that by only applying our GRASP algorithm and the SB, we already obtain very competitive gaps, usually better than the previous approaches except for SALRP on instances of set $\mathscr{F}_1$ and for SALRP and ALNS on instances of set $\mathscr{F}_3$.

Finally, in Table 5.X we report the new best known feasible solutions found by our algorithm. Note that these solutions were not necessarily found during the 10 runs of our method, but rather during the calibration of several parameters. In total, our algorithm was able to improve the solutions on 17 out of the 89 instances considered in this sutdy.

5.7 Concluding Remarks

In this paper we have introduced a new heuristic method for the CLRP based on a GRASP followed by the iterative solution of a new ILP model, the location-reallocation model (LRM). The GRASP introduced in this paper provides better solutions than the previous approach of Prins et al. [121] for most of the instances considered in this study. We have introduced the location-reallocation model that generalizes the CFLP and the RM of de Franceschi et al. [48] by simultaneously determining the locations of facilities as well as the reallocation of customers and routes to those facilities. We have introduced a new heuristic method, the solution blender (SB), that takes as input a set of solutions for the CLRP and solves the LRM to find near optimal solutions. Indeed, by only applying our GRASP followed by the SB we obtain gaps that are competitive with the methods found in the literature. We complement this by applying a local improvement heuristic based on the iterative solution of the LRM solved by column and cut generation. This local improvement heuristic was found to be very effective in tightening the optimality gap. Finally, we were able to improve the best known feasible solutions on 17 out of the 89 instances considered in this study.

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
ppw-20x5-1a	54793	54793	0.00	0.00	1.28	54793	0.00
ppw-20x5-1b	39104	39104	0.00	0.00	2.25	39104	0.00
ppw-20x5-2a	48908	48908	0.00	0.00	1.21	48908	0.00
ppw-20x5-2b	37542	37542	0.00	0.00	2.28	37542	0.00
ppw-50x5-1a	90111	90111	0.00	0.00	12.18	90111	0.00
ppw-50x5-1b	63242	63248	0.03	0.01	17.40	63242	0.00
ppw-50x5-2a	88298	88332	0.12	0.04	14.77	88298	0.00
ppw-50x5-2b	67308	67554	0.34	0.37	18.53	67373	0.10
ppw-50x5-2bis	84055	84055	0.00	0.00	17.72	84055	0.00
ppw-50x5-2bbis	51822	51898	0.02	0.15	24.06	51883	0.12
ppw-50x5-3a	86203	86203	0.00	0.00	14.76	86203	0.00
ppw-50x5-3b	61830	61836	0.03	0.01	20.16	61830	0.00
ppw-100x5-1a	274814	275626	0.06	0.30	188.51	275406	0.22
ppw-100x5-1b	213615	214699	0.12	0.51	178.81	214308	0.32
ppw-100x5-2a	193671	194118	0.17	0.23	106.96	193769	0.05
ppw-100x5-2b	157095	157238	0.05	0.09	94.29	157157	0.04
ppw-100x5-3a	200079	200341	0.02	0.13	86.76	200277	0.10
ppw-100x5-3b	152441	152737	0.26	0.19	95.87	152441	0.00
ppw-100x10-1a	287983	293117	2.79	1.78	1840.90	288415	0.15
ppw-100x10-1b	231763	233416	0.79	0.71	2329.90	**230989**	-0.33
ppw-100x10-2a	243590	244022	0.11	0.18	211.45	243695	0.04
ppw-100x10-2b	203988	204200	0.17	0.10	242.75	203988	0.00
ppw-100x10-3a	250882	252371	0.36	0.59	2576.34	250882	0.00
ppw-100x10-3b	204317	204996	0.14	0.33	1005.74	204602	0.14
ppw-200x10-1a	477248	476674	0.12	-0.12	3785.47	**475344**	-0.40
ppw-200x10-1b	378065	378781	0.27	0.19	3646.74	**377043**	-0.27
ppw-200x10-2a	449571	449469	0.05	-0.02	5215.70	**449152**	-0.09
ppw-200x10-2b	374330	375053	0.13	0.19	2831.53	374469	0.04
ppw-200x10-3a	469433	471218	0.13	0.38	4356.16	469706	0.06
ppw-200x10-3b	362817	363755	0.18	0.26	4936.13	**362743**	-0.02
Average			0.22	0.22	1129.22		0.01

Table 5.I: Results on instances of set $\mathscr{F}_1$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
P111112	1467.7	1475.4	0.24	0.52	171.94	1468.2	0.03
P111122	1449.2	1454.2	0.35	0.35	474.10	1449.2	0.00
P111212	1394.8	1405.0	0.36	0.73	161.80	1396.6	0.13
P111222	1432.3	1445.4	0.45	0.92	505.91	1432.9	0.04
P112112	1167.2	1178.3	0.15	0.95	225.19	1176.3	0.78
P112122	1102.2	1106.0	0.25	0.34	415.47	1102.8	0.05
P112212	791.7	796.9	0.50	0.67	196.60	791.9	0.03
P112222	728.3	728.4	0.03	0.02	370.49	728.3	0.00
P113112	1238.5	1241.9	0.21	0.28	224.21	1239.4	0.08
P113122	1245.3	1246.4	0.07	0.09	471.62	1245.5	0.02
P113212	902.3	902.5	0.06	0.02	177.30	902.3	0.00
P113222	1018.3	1019.6	0.11	0.13	496.25	1018.3	0.00
P131112	1866.8	1934.7	0.24	3.64	1073.37	1928.0	3.28
P131122	1823.5	1834.2	0.32	0.58	2020.47	**1823.2**	-0.02
P131212	1965.1	1978.2	0.34	0.66	781.64	1969.8	0.24
P131222	1796.5	1800.2	0.23	0.21	1646.47	**1792.8**	-0.20
P132112	1443.3	1452.5	0.26	0.64	757.08	1447.5	0.29
P132122	1434.6	1448.1	0.34	0.94	2863.10	1443.8	0.64
P132212	1204.4	1206.1	0.05	0.14	958.65	1204.9	0.04
P132222	931.0	932.3	0.07	0.15	2466.29	931.7	0.08
P133112	1694.2	1711.7	0.43	1.03	991.68	1700.3	0.36
P133122	1392.0	1401.7	0.07	0.70	2016.18	1400.1	0.58
P133212	1198.3	1200.5	0.12	0.19	895.12	**1198.2**	-0.01
P133222	1151.8	1159.0	0.08	0.62	2640.94	1157.7	0.51
P121112	2251.9	2258.8	0.27	0.30	2094.06	**2249.0**	-0.13
P121122	2159.9	2161.4	0.18	0.07	4911.06	**2153.8**	-0.28
P121212	2220.0	2223.9	0.35	0.17	2304.13	**2212.4**	-0.34
P121222	2230.9	2238.6	0.17	0.34	5175.85	2232.5	0.07
P122112	2073.7	2094.5	0.31	1.00	3520.46	2085.0	0.54
P122122	1692.2	1709.0	0.24	1.00	7177.74	1703.8	0.69
P122212	1453.2	1469.2	0.18	1.10	4162.82	1465.9	0.87
P122222	1082.7	1087.2	0.17	0.41	7194.32	1083.9	0.11
P123112	1960.3	1971.7	0.23	0.58	3060.73	1966.7	0.33
P123122	1918.9	1941.6	0.23	1.18	9341.61	1932.7	0.72
P123212	1762.0	1769.8	0.18	0.44	3813.81	1765.8	0.22
P123222	1391.7	1393.9	0.11	0.16	5422.38	1392.4	0.05
Average			0.22	0.59	2255.02		0.27

Table 5.II: Results on instances of set $\mathscr{F}_2$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
Perl-12x2	203.98	203.98	0.00	0.00	0.16	203.98	0.00
Gas-21x5	424.90	424.90	0.00	0.00	1.24	424.90	0.00
Gas-22x5	585.11	585.11	0.00	0.00	2.54	585.11	0.00
Min-27x5	3062.02	3062.02	0.00	0.00	2.68	3062.02	0.00
Gas-29x5	512.10	512.10	0.00	0.00	4.53	512.10	0.00
Gas-32x5	562.22	562.25	0.03	0.00	5.04	562.22	0.00
Gas-32x5b	504.33	504.33	0.00	0.00	6.05	504.33	0.00
Gas-36x5	460.37	460.37	0.00	0.00	6.91	460.37	0.00
Chr-50x5ba	565.62	575.60	3.14	1.76	14.13	570.03	0.78
Chr-50x5be	565.60	580.98	10.20	2.72	15.26	565.60	0.00
Perl-55x15	1112.06	1112.66	0.54	0.05	42.14	1112.32	0.02
Chr-75x10ba	844.40	848.34	2.71	0.47	74.14	844.40	0.00
Chr-75x10be	848.85	853.87	1.38	0.59	84.97	850.93	0.24
Chr-75x10bmw	802.08	809.78	3.16	0.96	86.13	803.10	0.13
Perl-85x7	1622.50	1626.01	1.26	0.22	65.78	1623.86	0.08
Das-88x8	355.78	356.12	0.44	0.09	164.40	355.78	0.00
Chr-100x10	833.43	851.00	6.47	2.11	350.88	841.68	0.99
Min-134x8	5709.00	5816.73	66.78	1.89	1188.96	5719.25	0.18
Das-150x10	43963.60	44321.33	83.83	0.81	1311.31	44179.00	0.49
Average			9.47	0.61	180.38		0.15

Table 5.III: Results on instances of set $\mathscr{F}_3$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
M-n150x14a	1352.93	1354.73	1.38	0.13	1089.83	1353.46	0.04
M-n150x14b	1212.46	1219.44	4.16	0.58	942.44	1215.14	0.22
M-n199x14a	1644.35	1645.97	1.74	0.10	3107.52	1644.35	0.00
M-n199x14b	1480.43	1488.37	2.53	0.54	3050.01	1483.55	0.21
Average			2.45	0.34	2047.45		0.12

Table 5.IV: Results on instances of set $\mathscr{F}_4$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
ppw-20x5	0.03	0.42	0.00	0.53	0.00	0.68	0.00	1.00	0.00	1.75
ppw-50x5	0.70	7.76	0.14	8.58	0.12	10.34	0.08	12.93	0.07	17.45
ppw-100x5	1.52	67.56	0.38	81.93	0.35	88.86	0.30	100.19	0.24	125.20
ppw-100x10	3.51	78.92	2.33	376.89	1.01	815.67	0.81	1033.83	0.62	1367.85
ppw-200x10	1.57	1036.77	0.60	1704.85	0.25	2633.61	0.19	3133.39	0.15	4128.62
Average	1.51	238.78	0.70	435.09	0.35	710.48	0.28	857.06	0.22	1129.22

Table 5.V: Algorithm evolution for instances of set $\mathscr{F}_1$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
100x10	2.24	93.71	0.72	100.64	0.64	118.31	0.56	146.15	0.53	192.84
100x20	1.82	147.23	0.40	165.46	0.36	220.71	0.34	297.23	0.31	455.64
150x10	3.11	490.65	1.24	543.03	1.13	626.82	1.10	716.21	1.05	909.59
150x20	2.71	678.91	0.67	771.96	0.60	1202.04	0.57	1533.36	0.53	2275.57
200x10	3.29	1591.95	1.02	1836.23	0.81	2409.46	0.71	2602.60	0.60	3159.34
200x20	3.63	2253.24	0.71	2540.08	0.59	4072.30	0.57	4732.70	0.53	6537.16
Average	2.80	875.95	0.79	992.90	0.69	1441.61	0.64	1671.38	0.59	2255.02

Table 5.VI: Algorithm evolution for instances of set $\mathscr{F}_2$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
≤ 50 custs	1.18	2.41	0.65	2.56	0.57	3.06	0.50	4.13	0.45	5.85
> 50 custs	2.93	91.68	1.21	97.85	1.01	144.40	0.88	221.50	0.80	374.30
Average	2.01	44.69	0.91	47.70	0.78	70.01	0.68	107.09	0.61	180.38

Table 5.VII: Algorithm evolution for instances of set $\mathscr{F}_3$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
150 custs	5.42	677.23	0.43	759.88	0.40	817.56	0.36	892.03	0.35	1016.14
199 custs	6.15	2078.87	0.44	2362.29	0.35	2530.13	0.33	2689.19	0.32	3078.77
Average	5.78	1378.05	0.44	1561.08	0.38	1673.85	0.35	1790.61	0.34	2047.45

Table 5.VIII: Algorithm evolution for instances of set $\mathscr{F}_4$

Method	$\mathscr{F}_1$		$\mathscr{F}_2$		$\mathscr{F}_3^{\S}$	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
GRASP	3.60	96.5	3.42	159.56	1.49	21.15
MA|PM	1.38	76.7	1.78	203.13	2.01	37.8
LRGTS	0.74	17.5	1.76	21.24	1.64	18.21
VNS+ILP	0.86	6.7	–	–	–	–
SALRP	0.41	422.4	1.41	826.4	0.27	140.46
ALNS	0.70	451.0	0.81	830.0	0.15	174.75
GRASP†	1.51	238.78	2.80	875.95	1.90	54.87
GRASP+SB‡	0.70	435.09	0.79	992.90	1.01	57.74
GRASP+ILP	0.22	1129.22	0.59	2255.02	0.64	254.98

§ Subset of instances considered by all methods reporting results for $\mathscr{F}_3$

† Only considering the first stage of our method

‡ Only considering the first and second stages of our method

Table 5.IX: Comparison of average results

Instance	z^*_{BKS}	z^*_{NEW}	Instance	z^*_{BKS}	z^*_{NEW}
ppw-100x10-1a	287983	**287695**	P111222	1432.3	**1432.2**
ppw-100x10-1b	231763	**230989**	P113212	902.3	**902.2**
ppw-200x10-1a	477248	**475294**	P113222	1018.3	**1018.2**
ppw-200x10-1b	378065	**377043**	P131122	1823.5	**1823.2**
ppw-200x10-2a	449571	**449115**	P131222	1796.5	**1792.7**
ppw-200x10-2b	374330	**374280**	P133212	1198.3	**1198.2**
ppw-200x10-3b	362817	**362653**	P121112	2251.9	**2248.9**
			P121122	2159.9	**2153.8**
			P121212	2220.0	**2212.4**
			P121222	2230.9	**2222.9**

Table 5.X: New best known solutions

CHAPTER 6

CONCLUSIONS

This dissertation has addressed the capacitated location-routing problem (CLRP), an important logistics problem combining operational with tactical and strategic planning, which arises in many real-life applications, such as the location of warehouses or the operation of city logistics systems [43]. The CLRP is an $\mathscr{NP}$-hard problem as it combines two other well known $\mathscr{NP}$-hard problems, the CFLP and the CVRP. In the CLRP, a planner must decide the locations of a series of facilities and schedule vehicle routes so as to serve the demand of a known set of customers, at minimum cost. It is assumed that the vehicle fleet is homogeneous and that the network is symmetric. Therefore, the CLRP is also a simplification of the location-routing problem (LRP) in which some of these assumptions are relaxed. Recent contributions to the solution of the CLRP exploit these homogeneities and symmetries to derive compact models and tight formulations which allow the exact solution of small and medium size instances. In this dissertation, we also make use of such assumptions so as to derive stronger models and faster, more efficient algorithms to solve the CLRP either exactly or heuristically in reasonable computing times.

In Chapter 3 we introduce three new formulations for the CLRP, based on vehicle and commodity flows. These formulations are shown to dominate the original two-index vehicle-flow formulation of Belenguer et al. [19] in terms of the linear relaxation lower bound. We introduce several new families of valid inequalities, strengthen some of the existing ones, and present efficient separation algorithms that in many cases generalize the separation procedures introduced by Belenguer et al. [19]. We compare the different flow formulations on a large set of instances, concluding that compact two-index vehicle-flow formulations are more efficient to handle the small and medium size instances considered in our study. However, three-index formulations are able to obtain better gaps on some instances and to scale better than two-index formulations, which suggests that important information about the structure of the optimal solutions of the

CLRP is lost or poorly represented in compact two-index formulations. We also compare our implementation of the branch-and-cut algorithm on the two-index vehicle-flow formulation against the method presented by Belenguer et al. [19]. Our method is able to solve instances with up to 100 customers, whereas the original method of Belenguer et al. [19] solves instances with up to 50 customers. Several refinements in our implementation explain these results, including the new valid inequalities, the efficient separation algorithms as well as the separation strategy used in the branch-and-cut algorithm which deactivates cuts that do not seem promising in a certain branch of the tree.

In Chapter 4 we introduce a branch-and-cut-and-price algorithm for the CLRP. We consider the set-partitioning formulation of the problem for which we show how to incorporate all the valid inequalities from the flow formulations introduced in the Chapter 3. We introduce five new families of valid inequalities which are shown to dominate some of the inequalities introduced before that are specific for the set-partitioning formulation of the CLRP. These new valid inequalities provide strong lower bounds. For the pricing problems, we use the shortest path problem with resource constraints and without cycles of length two or less (2-cyc-SPPRC), which normally provides lower bounds that are weaker than the ones obtained when pricing on elementary routes (routes without cycles). We propose a new family of degree constraints which force elementarity, allowing us to obtain lower bounds close to the ones obtained when pricing on elementary routes while still using 2-cyc-SPPRC as pricing algorithm. This is complemented with the use of new fathoming rules and specific dominance rules so as to, among other things, weaken the dominance with respect to the solution of the SPPRC with elementary routes (ESPPRC). These refinements have allowed the solution of some large instances of the CLRP containing up to 200 customers and improved its robustness by allowing the solution of all instances containing strictly less than 100 customers. We compare our algorithm against the state-of-the-art exact solvers for the CLRP, namely the branch-and-cut algorithms of Belenguer et al. [19] and the one over the two-index vehicle-flow formulation introduced in Chapter 3, and the branch-and-cut-and-price algorithm of Baldacci et al. [16]. We can conclude that our algorithm outperforms the branch-and-cut methods and is competitive against the branch-and-cut-and-price method of Baldacci et al. [16].

In particular, we are able to solve all instances also solved by the method of Baldacci et al. [16] plus four previously open instances, and we improve the best known solutions for three more instances.

In Chapter 5 we introduce a new heuristic for the CLRP based on the sequential application of a GRASP complemented with local search and followed by the solution of a series of integer-linear programs (ILP). These ILPs are based on the solution of a new model, the location-reallocation model (LRM) which generalizes the CFLP and the reallocation model of de Franceschi et al. [48], the first by allowing the reallocation of customers in the middle of routes, and the second by allowing the reallocation of routes to different facilities. This model can also be seen as a constrained CLRP and also inherits the cuts valid for the three-index formulations. In particular, we strengthen this model by including some of the inequalities described in Chapter 3. We apply a simple but efficient pricing algorithm and local branching strategies to solve the model quickly. We propose two heuristics that are based on the solution of the LRM. A solution blender (SB) heuristic takes the solutions available in a solution pool (initially we consider the solutions obtained by our GRASP). Each of these solutions is represented by a set of routes, and for each of the routes belonging to a solution, a replication step is used to build another set of routes connected to each possible facility. The LRM is then solved using as the initial set of columns the ones constructed during the replication step, and without applying any column generation. The other heuristic method, also based on the solution of the LRM, is what we call the local improvement heuristic (LIH). In the LIH, a destroy operator is applied to a solution to remove from it a set of customers. The destroyed solution is then repaired by solving the LRM, obtaining an eventually new solution of better quality. Our heuristic method is shown to be competitive against the existing methods. In particular, we obtain tighter average gaps than several of these methods on a large set of instances, which show the efficiency of our heuristic.

As a general conclusion, this dissertation addresses the CLRP by proposing new models and algorithms to solve it either exactly or heuristically, based on both pure metaheuristics and mathematical programming techniques. The algorithms introduced in this dissertation generalize existing methods in several respects. Moreover, our so-

lution approaches include exact and heuristic algorithms, and depending on the needs of a decision maker, some of them may be better suited in particular applications. Remarkably, flow formulations seem to be the right choice for solving small or medium size instances with few customers (normally less than 50) and few facilities (less than 10), while column generation-based algorithms scale better but their efficiency strongly depends on the quality of the lower bounds obtained as well as the size of the solution space in the dynamic programming pricing algorithms. Indeed, these methods usually perform very well when capacities are tight and feasible routes visit a small number of customers. In presence of loose capacities, the pricing algorithms require excessive computing time. Finally, a heuristic method may suit in most cases the needs of a planner, for whom obtaining the optimal solution may not be relevant, but rather a solution of good quality. The heuristic method introduced in this dissertation is shown to be one of the most robust and reliable in the literature, providing the tightest gaps on average in reasonable computing times.

BIBLIOGRAPHY

[1] K. Aardal. *On The Solution Of One And Two-Level Capacitated Facility Location Problems By The Cutting Plane Approach*. PhD thesis, Université Catholique de Louvain, Belgium, 1992.

[2] K. Aardal, Y. Pochet, and L. A. Wolsey. Capacitated facility location: Valid inequalities and facets. *Mathematics of Operations Research*, 20:562–582, 1995.

[3] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutella. A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50:749–760, 2004.

[4] Z. Akca, R. T. Berger, and T. K. Ralphs. Modeling and solving location, routing, and scheduling problems. In *Proceedings of the Eleventh INFORMS Computing Society Meeting*, pages 309–330, Charleston, South Carolina, USA, 2009.

[5] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, 2007.

[6] J. Araque, L. Hall, and T. Magnanti. Capacitated trees, capacitated routing, and associated polyhedra. CORE discussion paper, Université Catholique de Louvain, Belgium, 1990.

[7] C. Archetti, M. Bouchard, and G. Desaulniers. Enhanced branch-and-price-and-cut for vehicle routing with split deliveries and time windows. Technical report, Cahiers du GERAD, Montréal, Canada, 2009.

[8] P. Augerat. *Approche polyhédrale du problème de tournées de véhicules*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1995.

[9] I. Averbakh and O. Berman. Routing and location-routing p-delivery men problems on a path. *Transportation Science*, 28:162–166, 1994.

[10] I. Averbakh and O. Berman. Minmax p-traveling salesmen location problems on a tree. *Annals of Operations Research*, 110:55–68, Feb. 2002.

[11] E. Balas and M. W. Padberg. Set-partitioning: a survey. *SIAM Review*, 18:710–760, 1976.

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

[13] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52:723–738, 2004.

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

[15] R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7:229–268, 2010.

[16] R. Baldacci, A. Mingozzi, and R. Wolfler-Calvo. An exact method for the capacitated location-routing problem. *Operations Research*, 2011. Forthcoming.

[17] J. Barceló and J. Casanovas. A heuristic Lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15:212–226, 1984.

[18] S. Barreto. *Análise e Modelização de Problemas de localização-distribuição*. PhD thesis, University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal. In Portuguese, 2004.

[19] J. M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler-Calvo. A branch-and-cut algorithm for the capacitated location routing problem. *Computers & Operations Research*, 38:931–941, 2010.

[20] M. Bellmore and G. L. Nemhauser. The traveling salesman problem: A survey. *Operations Research*, 16:538–558, 1968.

[21] A. Billionet, S. Elloumi, and L. Grouz Djerbi. Designing radio-mobile access networks based on synchronous digital hierarchy rings. *Computers & Operations Research*, 32:379–394, 2005.

[22] U. Blasum and W. Hochstättler. Application of the branch and cut method to the vehicle routing problem. Technical report, Zentrum fur Angewandte Informatik Köln, Germany, 2002.

[23] B. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34:58–68, 2006.

[24] S. C. Boyd and W. H. Cunningham. Small travelling salesman polytopes. *Mathematics of Operations Research*, 16:259–259, 1991.

[25] J. Bramel and D. Simchi-Levi. A location based heuristic for general routing problems. *Operations Research*, 43:649–660, 1995.

[26] P. Cappanera, G. Gallo, and F. Maffioli. Discrete facility location and routing of obnoxious activities. *Discrete Applied Mathematics*, 133:3–28, 2004.

[27] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33:2972–2990, 2006.

[28] D. C. Cho, E. L. Johnson, M. Padberg, and M. R. Rao. On the uncapacitated plant location problem. I: Valid inequalities and facets. *Mathematics of Operations Research*, 8:579–589, 1983.

[29] D. C. Cho, M. Padberg, and M. R. Rao. On the uncapacitated plant location problem. II: Facets and lifting theorems. *Mathematics of Operations Research*, 8: 590–612, 1983.

[30] N. Christofides and J. E. Beasley. An algorithm for the capacitated warehouse location problem. *European Journal of Operational Research*, 12:19–28, 1983.

[31] N. Christofides, A. Mingozzi, and P. Toth. *The Vehicle Routing Problem*. Wiley, Chichester, UK, 1979.

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

[33] V. Chvátal. Edmonds polytopes and weakly Hamiltonian graphs. *Mathematical Programming*, 5:29–40, 1973.

[34] V. Chvátal, D. Applegate, R. Bixby, and W. Cook. CONCORDE: A code for solving traveling salesman problems, 1999. URL `http://www.tsp.gatech.edu/concorde.html`.

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

[36] C. Contardo, J.-F. Cordeau, and B. Gendron. A computational comparison of flow formulations for the capacitated location-routing problem. Technical Report CIRRELT-2011-47, Université de Montréal, 2011.

[37] C. Contardo, J.-F. Cordeau, and B. Gendron. A branch-and-cut-and-price algorithm for the capacitated location-routing problem. Technical Report CIRRELT-2011-44, Université de Montréal, 2011.

[38] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problem. *Networks*, 30:105–119, 1997.

[39] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53: 512–22, 2002.

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

[41] G. Cornuéjols and J.-M. Thizy. Some facets of the simple plant location polytope. *Mathematical Programming*, 23:50–74, 1982.

[42] G. Cornuéjols, J. Fonlupt, and D. Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33:1–27, 1985.

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

[44] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6:791–812, 1958.

[45] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[46] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, 2:393–410, 1954.

[47] M. Daskin. What you should know about location modeling. *Naval Research Logistics*, 55:283–294, 2008.

[48] R. de Franceschi, M. Fischetti, and P. Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105:471–499, 2006.

[49] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42:387–404, 2008.

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

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

[52] J. A. Díaz and E. Fernández. A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*, 53:728–740, 2002.

[53] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

[54] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009, 1978.

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

[56] D. Feillet, M. Gendreau, and L.-M. Rousseau. New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, 45:239–256, 2007.

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

[58] B. Fleischmann. A new class of cutting planes for the symmetric travelling salesman problem. *Mathematical Programming*, 40:225–46, 1988.

[59] R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming Series A*, 106:491–511, 2006.

[60] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[61] R. S. Garfinkel, A. Neebe, and M. Rao. An algorithm for the m-median plant location problem. *Transportation Science*, 8:848–856, 1974.

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

[63] A. M. Geoffrion and R. McBride. A lagrangian relaxation applied to capacitated facility location problems. *AIIE Transactions*, 10:40–47, 1978.

[64] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[65] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances And New Challenges; Electronic Version*. Operations Research/Computer Science Interfaces. Springer, Dordrecht, 2008.

[66] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen. Implementing vehicle routing problems. *Networks*, 7:113–148, 1977.

[67] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the SIAM*, 9:551–570, 1961.

[68] M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem I: inequalities. *Mathematical Programming*, 16:265–280, 1979.

[69] M. Grötschel and W. Pulleyblank. Clique tree inequalities and the symmetric travelling salesman problem. *Mathematics of Operations Research*, 11:537–569, 1986.

[70] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: computation. *INFORMS Journal on Computing*, 10: 427–437, 1998.

[71] H. Gunnarsson, M. Rönnqvist, and D. Carlsson. A combined terminal location and ship routing problem. *Journal of the Operational Research Society*, 57:928–938, 2006.

[72] G. Gutin and A. P. Punnen, editors. *The Traveling Salesman Problem And Its Variations*. Combinatorial Optimization. Springer, 2007.

[73] S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12:450–459, 1964.

[74] S. L. Hakimi. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13: 462, 1965.

[75] P. Hansen and N. Mladenovic. Variable neighborhood search for the *p*-median. *Location Science*, 5:207–226, 1997.

[76] P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[77] P. H. Hansen, B. Hegedahl, S. Hjortkjaer, and B. Obel. A heuristic solution to the warehouse location-routing problem. *European Journal of Operational Research*, 76:111–127, 1994.

[78] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical Report CIRRELT-2011-42, Université de Montréal, 2011.

[79] K. Holmberg, M. Rönnqvist, and D. Yuan. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113:544–559, 1999.

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

[81] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. I: The *p*-centers. *SIAM Journal on Applied Mathematics*, 37:513–538, 1979.

[82] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. II: The *p*-medians. *SIAM Journal on Applied Mathematics*, 37:539–560, 1979.

[83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[84] J. G. Klincewicz and H. Luss. A lagrangian relaxation heuristic for capacitated facility location with single source constraints. *Journal of the Operational Research Society*, 37:495–500, 1986.

[85] A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9:643–666, 1963.

[86] M. Labbé and G. Laporte. Maximizing user convenience and postal service efficiencyin post box location. *Journal of Operational Research, Statistics and Computer Science*, 26, 1986.

[87] G. Laporte. Location-routing problems. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*. Elsevier Science Publishers B.V. (North-Holland), 1988.

[88] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43:408–416, 2009.

[89] G. Laporte and Y. Nobert. An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6: 224–226, 1981.

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

[91] G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6:293–310, 1986.

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

[93] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, Chichester, 1985.

[94] A. Letchford. Separating a superclass of comb inequalities in planar graphs. *Mathematics of Operations Research*, 25:443–454, 2000.

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

[96] J. M. Y. Leung and T. L. Magnanti. Valid inequalities and facets of the capacitated plant location problem. *Mathematical Programming*, 44:271–291, 1989.

[97] C. K. Y. Lin, C. K. Chow, and A. Chen. A location-routing-loading problem for bill delivery services. *Computers & Operations Research*, 43:5–25, 2002.

[98] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[99] S. C. Liu and C. C. Lin. A heuristic method for the combined location routing and inventory problem. *International Journal of Advanced Manufacturing Technologies*, 26:372–381, 2005.

[100] J. Lysgaard. URL `http://www.hha.dk/~lys`.

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

[102] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45:414–424, March 1999.

[103] S. T. McCormick, M. R. Rao, and G. Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming Series B*, 94:459–466, 2003.

[104] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[105] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7:326–329, 1960.

[106] N. Mladenovic, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez. The *p*-median problem: a survey of metaheuristic approaches. *European Journal of Operational Research*, 179:927–939, 2007.

[107] R. H. Mole and S. R. Jameson. A sequential route-building algorithm employing a generalized savings criterion. *Operations Research Quarterly*, 27:503–511, 1976.

[108] D. Naddef and G. Rinaldi. The symmetric traveling salesman polytope and its graphical relaxation: composition of valid inequalities. *Mathematical Programming*, 51:359–400, 1991.

[109] D. Naddef and G. Rinaldi. The graphical relaxation: a new framework for the symmetric traveling salesman polytope. *Mathematical Programming*, 58:53–88, 1993.

[110] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177:649–672, 2007.

[111] A. W. Neebe and M. R. Rao. An algorithm for the fixed-charge assigning users to sources problem. *Journal of the Operational Research Society*, 34:1107–1113, 1983.

[112] I. Or and W. P. Pierskalla. A transportation location-allocation model for the regional blood banking. *AIIE Transactions*, 11:86–95, 1979.

[113] Z. Ozyurt and D. Aksen. Solving multi-depot location-routing problem with time windows using lagrangian relaxation. In *ODYSSEUS 2006. Third International Workshop on Freight Transportation and Logistics*, Altea, Spain, 2006.

[114] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ, USA, 1982.

[115] J. Perl and M. S. Daskin. A warehouse location-routing problem. *Transportation Research B*, 19B:381–396, 1985.

[116] S. Pirkwieser and G. R. Raidl. Variable neighborhood search coupled with ILP-based very large-neighborhood searches for the (periodic) location-routing problem. In *Hybrid Metaheuristics - Seventh International Workshop, HM 2010*, volume 6373 of *Lecture Notes in Computer Science*, pages 174–189, Vienna, 2010.

[117] D. Pisinger and S. Røpke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[118] Y. Pochet and L. A. Wolsey. Lot-sizing with constant batches: formulation and valid inequalities. *Mathematics of Operations Research*, 18:767–785, 1993.

[119] J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routeing problems with time windows. *The Journal of the Operational Research Society*, 46:1433–1446, 1995.

[120] C. Prins, C. Prodhon, and R. Wolfler-Calvo. A memetic algorithm with population management (MA|PM) for the capacitated location-routing problem. In J. Gottlieb and G. R. Raidl, editors, *EvoCOP 2006, Lecture Notes in Computer Science*, volume 3906, pages 183–194. Springer-Verlag Berlin Heidelberg, 2006.

[121] C. Prins, C. Prodhon, and R. Wolfler-Calvo. Solving the capacitated location-routing problem by a GRASP complemented by a learning process and path relinking. *4OR*, 4:221–238, 2006.

[122] C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfler-Calvo. Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41:470–483, 2007.

[123] C. Prodhon. *Le problème de localisation-routage*. PhD thesis, Université de Technologie de Troyes, France, 2006.

[124] C. Prodhon. An ELS x path relinking hybrid for the periodic location-routing problem. In M. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, and A. Schaerf, editors, *Hybrid Metaheuristics*, volume 5818 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin / Heidelberg, 2009.

[125] C. Prodhon. A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal of Operational Research*, 210:204–212, 2011.

[126] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, Berlin, Heidelberg, 1994.

[127] C. Revelle, H. Eiselt, and M. Daskin. A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184:817–848, 2008.

[128] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51:155–170, 2008.

[129] S. Røpke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40: 455–472, 2006.

[130] S. Salhi and G. Nagy. Local improvement in planar facility location using vehicle routing. *Annals of Operations Research*, 167:287–296, 2009.

[131] S. Salhi and G. K. Rand. Effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39:150–156, 1989.

[132] S. Salhi, M. Sari, D. Saidi, and N. Touati. Adaptation of some vehicle fleet mix heuristics. *Omega*, 20:653–660, 1992.

[133] M. Schwardt and J. Dethloff. Solving a continuous location-routing problem by use of a self-organizing map. *International Journal of Physical Distribution & Logistics Management*, 35:390–408, 2005.

[134] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

[135] D. Simchi-Levi. The capacitated traveling salesmen location problem. *Transportation Science*, 25:9–18, 1991.

[136] H. K. Smith, G. Laporte, and P. R. Harper. Locational analysis: Highlights of growth to maturity. *Journal of the Operational Research Society*, 60:140–148, 2009.

[137] K. Sörensen and M. Sevaux. MAPM: memetic algorithms with population management. *Computers & Operations Research*, 33:1214–1225, 2006.

[138] B. Srikar and R. Srivastava. Solution methodology for the location routing problem. In *Proceedings of the II ORSA/TIMS Conference*, Orlando, FL., 1983.

[139] M. B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16:955, 1968.

[140] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[141] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15:333–46, 2003.

[142] D. Tuzun and L. I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116:87–99, 1999.

[143] T. J. Van Roy. A cross decomposition algorithm for capacitated facility location. *Operations Research*, 34:145–163, 1986.

[144] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Technical Report CIRRELT-2010-34, Université de Montréal, 2010.

[145] L. A. Wolsey. Submodularity and valid inequalities in capacitated fixed charge networks. *Operations Research Letters*, 8:119–124, 1989.

[146] T.-H. Wu, C. Low, and J.-W. Bai. Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29:1393–1415, 2002.

[147] V. F. Yu, S.-W. Lin, W. Lee, and C.-J. Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, 58:288–299, 2010.